MULTI-ROBOT ASSIGNMENT AND FORMATION CONTROL

A Thesis
Presented to
The Academic Faculty

by

Edward A. Macdonald

In Partial Fulfillment
of the Requirements for the Degree
Masters of Science in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
August 2011

MULTI-ROBOT ASSIGNMENT AND FORMATION CONTROL

Approved by:

Professor Magnus Egerstedt, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Ayanna Howard
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Jeff Shamma
School of Electrical and Computer Engineering
Georgia Institute of Technology

Date Approved: 7 July 2011

ACKNOWLEDGEMENTS

I want to thank my advisor, Magnus Egerstedt, who introduced me to the field of multi-agent robotics and whose support lead to the development of this thesis. I'd also like to thank Jean-Pierre de la Croix and Amir Rahmani for their assistance in setting up the experimental equipment. Lastly, I'd like to thank Jenna Barrows for her continued support of my decision to pursue a Master's degree. This thesis was partially supported by AFOSR project number: FA9550-09-1-0538.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS . . . . . **iii**

LIST OF TABLES . . . . . **vi**

LIST OF FIGURES . . . . . **vii**

SUMMARY . . . . . **ix**

I INTRODUCTION **1**

1.1 Multi-Agent Robotics 1

1.1.1 Formation Control 2

1.2 Goals for This Work 4

II BACKGROUND AND PREVIOUS WORK **5**

2.1 Graph Theory and Terminology 5

2.1.1 Edges and Vertices 5

2.1.2 Types of Graphs 6

2.2 Network Control Systems 6

2.2.1 Consensus Based Formation Control Methods 7

2.2.2 Distributed Assignment Algorithms 9

2.3 The Hungarian Algorithm 11

2.4 The Iterative Closest Point Algorithm 12

III THEORY **14**

3.1 Problem Definition 14

3.2 The Assignment Algorithm 14

3.2.1 Definition of the Assignment Algorithm 15

3.2.2 Convergence Proof 16

3.2.3 Distributed Implementation on Mobile Robot Systems 20

3.3 Variations on the Algorithm 25

3.3.1 The Non-Complete Graph Case 25

3.3.2 Leader Based Assignment and Formation Control 28

3.4 Comparisons With Previous Work 30
3.4.1 Required Information at the Robot Level 31
3.4.2 Efficiency 32
3.4.3 Robustness 32
3.4.4 Summary 33
IV SIMULATION 34
4.1 Complete Graph Case Simulations 34
4.1.1 Number of Nash Equilibria 38
4.2 Leader Based, Complete Graph Simulations 40
4.3 Non-Complete Graph Simulations 41
4.4 Summary 44
V EXPERIMENTATION 46
5.1 Experimental Setup 46
5.1.1 Equipment 46
5.1.2 Methods 49
5.2 Experimental Results 56
5.2.1 GRITS Formation Sequence 56
5.2.2 Leader Based Formation Control 58
5.3 Summary 61
VI CONCLUSION 63
6.1 Future Work 64
REFERENCES 66

LIST OF TABLES

1 Comparison of Formation Control Methods 33

LIST OF FIGURES

1 Examples of Graphs. Nodes are represented by circles, edges are black lines. 7

2 Examples of Rigid and Non-Rigid Graphs. Nodes are represented by circles, edges are black lines. 9

3 Number of Unique Assignments vs. Network Size, based on simulation with random data sets 19

4 Scenario with assignment of two robots. Robots are represented by two circles, and destination points shown by stars. Two possible assignments: Solid lines show assignment with crossing paths, dotted lines show assignment with no crossing paths. Letters denote the lengths of line segments. 24

5 Simulation data for a network of 8 robots moving to a box formation. Circles show the initial states, Xs show the final states, and lines show the paths taken. 35

6 Simulation data for a network of 15 robots moving to an arrow formation. Circles show the initial states, Xs show the final states, and lines show the paths taken. 36

7 Simulation data for a network of 29 robots moving to a "GT" formation. Circles show the initial states, Xs show the final states, and lines show the paths taken. 36

8 Simulation data for a network of 8 robots moving to a box formation. The '*' robot in the bottom center does not move. Other robots dynamically change the pose of the formation to compensate. 37

9 Simulation data for a network of 8 robots moving to a box formation. The '*' robot has a fixed assignment to a point on the edge. Other robots dynamically change pose and assignment to compensate. . . . 37

10 Simulation to compute the number of solutions vs. network size. 10 trials for each network size, N. Given that the centroids are aligned, the solid lines show the total number of optimal assignments. Number of Nash Equilibria shown with dotted lines. Means shown with circles. Maxes shown with +'s. Mins shown with triangles. 39

11 Simulation data for a network of 8 robots moving to a box formation with designated leader. The '*' is designated the leader and does not move. Other robots move in straight lines into formation. 40

12 Formation Synthesis over time with moving leader. Leader shown by '*' moves at constant velocity to the right. Robot paths shown by lines and positions at various times shown by x's. 41

13 Simulation data for a network of 8 robots moving to a box formation with a cycle graph. Robots start sufficiently close to the desired formation such that formation synthesis is successful. Graph edges shown by dotted lines. 42

14 Simulation data for a network of 8 robots moving to a box formation with a cycle graph. Robots do not start sufficiently close to the desired formation and fail to build the formation. Graph edges shown by dotted lines. 43

15 Simulation data for a network of 8 robots moving to a box formation with a cycle graph. Robots have fixed assignments and can recognize each other. Formation synthesis is successful for an arbitrary starting configuration. Graph edges shown by dotted lines. 44

16 Illustration of the Khepera III robot. 1: Infrared Sensors, 2: Ultrasonic Sensors, 3: Expansion slot for compact flash wireless card 47

17 Picture of a Khepera III robot with a mechanical pencil shown for scale. The reflective spheres on top of the robot are used in conjunction with the Vicon motion capture system. 48

18 Image of Vicon cameras overlooking a group of Khepera III robots. 3 cameras shown, 8 cameras total. 49

19 Software flow chart illustrating the program running on-board the Khepera robots. 50

20 Overhead view showing the orientation of the robot coordinate frames. 52

21 A video of 15 Khepera robots demonstrating the assignment algorithm using a sequence of formations to spell "G-R-I-T-S" (links to *macdonald-edward-a-201108-mast-formation-15.avi*) 57

22 A video of 5 Khepera robots demonstrating the leader version of the assignment algorithm using several geometric formations (links to *macdonald-edward-a-201108-mast-formation-leader.avi*) 60

SUMMARY

The fundamental goal of multi-agent robotics is simple: how to create control laws and behaviors that, when executed by each individual robot, some desirable global behavior emerges. The global behavior may range from something as simple as the robots meeting at a single point, to something as complex as a collective search and rescue mission.

Our research focuses on one of the more fundamental issues in multi-agent, mobile robotics: the formation control problem. The idea is to create controllers that cause robots to move into a predefined formation shape. This is a well studied problem for the scenario in which the robots know in advance to which point in the formation they are assigned. In our case, we assume this information is not given in advance, but must be determined dynamically. This thesis presents an algorithm that can be used by a network of mobile robots to simultaneously determine efficient robot assignments and formation pose for rotationally and translationally invariant formations. This allows simultaneous role assignment and formation sysnthesis without the need for additional control laws.

The thesis begins by introducing some general concepts regarding multi-agent robotics. Next, previous work and background information specific to the formation control and assignment problems are reviewed. Then the proposed assignment algorithm for role assignment and formation control is introduced and its theoretical properties are examined. This is followed by a discussion of simulation results. Lastly, experimental results are presented based on the implementation of the assignment algorithm on actual robots.

CHAPTER I

INTRODUCTION

With lowering cost and increasing computing power of embedded processors, multi-robot systems are beginning to emerge as viable solutions to real world problems. Since it is a relatively young field, there are many outstanding issues that must be confronted. This thesis presents an approach to one of these open problems: how to efficiently build and maintain formations using mobile robots. Our goal is to develop a method of dynamic role assignment and formation control of multi-robot systems for rotationally and translationally invariant formations.

This chapter introduces some of the basic concepts of multi-agent robotics and some of the challenges facing the field. An introduction into the formation control problem follows with a quick mention of some of the techniques previously used to approach the problem. The goals for this research are then introduced.

1.1 Multi-Agent Robotics

Network theory provides tools for researchers to analyze natural phenomena in a diverse number of fields such as social networks, sensor networks, biological systems, and material science. Using network theory, examination of the interactions that take place at the local level can provide insight into global system behavior. One such biological example is the study of schooling fish. Couzin et al used network theory to try to classify how fish, interacting at the local level, make group-level decisions without any clear single leader [7].

In addition to studying natural phenomena, network theory allows us to engineer networked systems that in some ways resemble their natural counterparts. Perhaps one of the most exciting areas where network theory has recently been applied is in

robotics. In an effort to reduce cost and complexity of autonomous systems, many researchers have turned to distributed solutions as opposed to centralized ones. The idea is that sometimes a single, highly complex, very expensive automated system is not well suited to solving a real world need. Centralized systems are well suited to tasks with a small scope such as CNC machines, robotic surgery, or serial manipulators. However, in instances where you have large and diverse environments, with tasks requiring the cooperation of multiple autonomous agents, networked robotic systems become essential. Tasks particularly well suited for multi-agent distributed systems include search and rescue type operations where a large area must be explored simultaneously, remote sensing and observation, and nano-robotics.

The primary challenge facing researchers and engineers in networked systems is identifying or creating rules that individuals within a network can follow that achieve some desirable global goal. Furthermore, these individuals often have limited access to only local information, and have limited computational ability. We know accomplishing this goal is possible through our observations of systems in nature such as ant colonies, or flocks of birds. It is through the study of these natural systems, synthesized within a mathematical framework, that engineers have begun to find solutions to these challenges.

1.1.1 Formation Control

Autonomous formation control is a desirable capability of many systems consisting of multiple mobile agents. Often times formation control is a necessary attribute of the system. For example, sensor arrays may need to maintain some formation in order to completely blanket an area with sensor coverage, or perhaps a spatial array formation is necessary to identify the direction of a propagating signal. Formations are also beneficial in groups of mobile robots moving to a position target. Only one robot needs to navigate, while the remaining robots can reach their destination

just by maintaining the formation. Furthermore, formation maintenance provides an efficient means of travel and prevents collisions between robots. With applications in areas such as mobile sensor coverage, unmanned aerial vehicle formation, satellite formation, and others, there has been a fair amount of interest in the subject.

Most work in this area has broken the formation control problem into two parts. The first is the assignment phase, where agents must determine which role to assume in the formation. Sometimes agents start out having their assignments predetermined, however this can lead to inefficiencies and a lack of robustness. If assignment is unknown, then each agent must find an assignment that doesn't conflict with other robots' assignments. For example, if robots simply drove to the closest point in the formation, multiple robots may end up driving to the same point. Furthermore it is desirable for robots to choose assignments that require as little travel as possible. If the formation has a fixed, known location, and the locations of all agents are also known, then the optimal assignment can be found using the Hungarian Algorithm developed by Harold Kuhn and James Munkres in the mid-twentieth century. If each agent doesn't have access to global information, then decentralized market driven, or auction algorithms have been proposed to determine efficient assignment[15]. Other attempts have robots drive to their closest formation point, and if occupied, simply move on to the next formation point[21]. All of these assignment algorithms require the formation location to be known in advance so robots can calculate the cost of a particular assignment. Little research has been conducted regarding role assignment if the location of the formation is not fixed or previously agreed upon.

The second aspect of the problem is how to achieve the desired formation after each robot's role has been assigned. Displacement based control using the consensus protocol is the primary method of formation control for translationally invariant formations if agents have a common orientation [9] [19]. Distance based formation control is most often used for translationally and rotationally invariant formations[3].

However, the distance based approach requires a rigid graph and introduces local minima, so agents must start close to the desired formation. In both of these methods, robots must know their assigned role, and be able to identify the assignments of robots around them.

1.2 Goals for This Work

The primary goal for this research is to develop a method of dynamic role assignment and formation control for multi-robot systems for rotationally and translationally invariant formations. As described above, most previous work has treated assignment and formation synthesis as two separate problems. Our approach is to determine formation pose as part of the assignment algorithm. This means robots will dynamically determine the formation pose in addition to role assignment. This will allow us to apply techniques such as the Hungarian algorithm even if the formation pose is not known in advance. Furthermore, determining the formation pose during the assignment phase yields a unique point in space to which each robot is assigned. This way, formation synthesis and assignment are accomplished simultaneously without having to introduce additional control laws such as distance based formation control.

The algorithm is developed from a theoretical standpoint for the case where each robot has access to complete network information. Stability and convergence properties are derived and discussed. Additionally, techniques for effective implementation on groups of mobile robots are introduced. This is then extrapolated to the case where robots only have access to local information. Simulations are used to verify and supplement the theory where applicable. Lastly, the algorithm has been implemented on a team of mobile Khepera robots. The experimental setup, methods, and results are given in detail.

CHAPTER II

BACKGROUND AND PREVIOUS WORK

As a way of representing the interaction between agents in a networked system, graph theory has become essential in understanding how local interactions affect global system behavior. This chapter begins with a brief introduction into some basic concepts and terminology of graph theory. This is by no means a thorough review, and only concepts required in the context of this thesis are covered. This is followed by the application of graph theory to networked systems, focusing on one of the most significant results of the past decade: the consensus equation. Deceptively simple, the consensus equation forms the basis for many methods used in network control. The two most well known methods of formation control, which are based upon the consensus equation, are then presented. Next, three examples of existing methods of role assignment within networks are reviewed. Lastly, the Iterative Closest Point Algorithm, which was the main inspiration for this work, is introduced.

2.1 Graph Theory and Terminology

2.1.1 Edges and Vertices

The fundamental purpose of graphs is to give a mathematical representation to the structure of a network. Graphs are made of two fundamental objects: edges and vertices. Vertices, also known as nodes or agents, are typically treated as points in space to represent the state of some object. For example, in multi-robot systems the vertices are the robots themselves and the locations of the vertices correspond to the robots' positions. Edges connect vertices and represent the flow of information from one vertex to another. However, edges tell us nothing about what type of information is being transferred. In the case of multi-robot systems, a graph edge often represents

the fact that two robots can observe each other's position. A graph is defined by its vertex set, and what edges exist connecting those vertices. If two vertices are connected by an edge, those two vertices are said to be "Neighbors". The neighbor set of a single vertex is the set of all other vertices which connect to that vertex with an edge. Figure 1a shows an example of a graph.

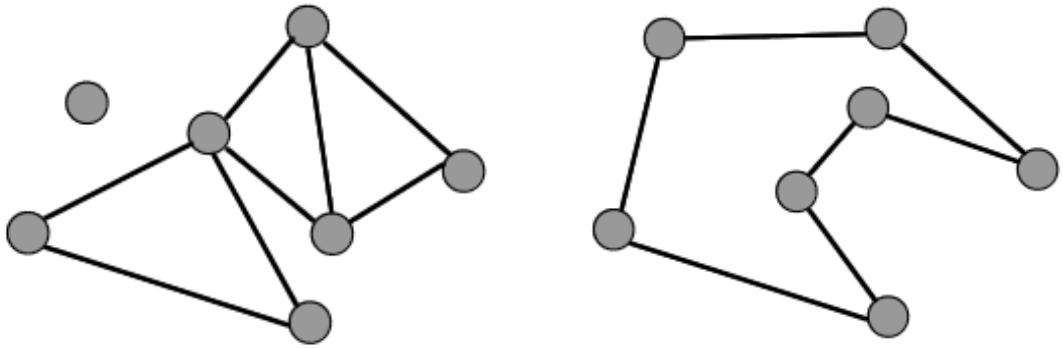
2.1.2 Types of Graphs

Given a certain number of vertices, there are many different types of graphs possible, depending on the connectivity of the network. A network that has a high connectivity is said to be very dense, and there are many edges connecting vertices. In this case each vertex has access to a lot of information about the network. The extreme case is called a complete graph, where there exists an edge between any two vertices in the network. In this case each vertex receives complete information about the state of a network. If a network has low connectivity, it is said to be sparse and there are relatively few edges connecting the vertices.

A graph is said to be connected if, by tracing along edges, a path exists between any two vertices in the graph. Having a graph that is connected is important for the consensus equation, which is discussed in the next section. Additionally, there are a few different types of canonical network topologies that often arise. One of which is the cycle graph, where every node has two edges and there are at least two paths from any node to any other node. Figure 1b shows an example of a cycle graph.

2.2 Network Control Systems

At the heart of networked control system study is the consensus protocol. The consensus protocol gives agents in a distributed network a means of "agreeing" upon a particular state even though there is no direct communication between all agents. Initially, each agent may start with a unique state value, but by using the consensus protocol, eventually all agents will converge to the same state provided the graph is



(a) An Example Graph (b) A Cycle Graph

Figure 1: Examples of Graphs. Nodes are represented by circles, edges are black lines.

connected[19]. The consensus equation is given by equation 1. x_i is the state of agent i, and $j \epsilon N_i$ denotes all agents, j, in the neighbor set of i.

$$\dot{x}_i(t) = \sum_{j \epsilon N_i} \big(x_j(t) - x_i(t)\big) \tag{1}$$

Consider, for example, a group of mobile robots randomly distributed along a line. Each robot can observe the positions of the neighboring robots. It is desirable for all robots to meet at the same point, or in other words, agree upon their position state value. If each robot updated its position according to the consensus equation, it can be shown that all robots will eventually meet at the same point provided the graph is connected. This not only works along a line, but can be extrapolated to any number of dimensions. The fact that this is a linear equation allows the use of linear system theory in analysis, greatly simplifying the complexity of networked systems.

2.2.1 Consensus Based Formation Control Methods

Two methods have emerged as the dominant formation control strategies for distributed systems: displacement based control, and distance based control [18]. Both strategies make use of the consensus equation. Furthermore, both strategies assume

that agents have a fixed role assignment, and that agents know the role assignments of their neighbors. In other words, agents must be able to recognize who their neighbors are. Displacement based formation control can be used to build translationally invariant formations, i.e. formations that can have any translation, but have a fixed rotation. A slight, linear, modification can be made to the consensus equation[14], as shown in equation 2.

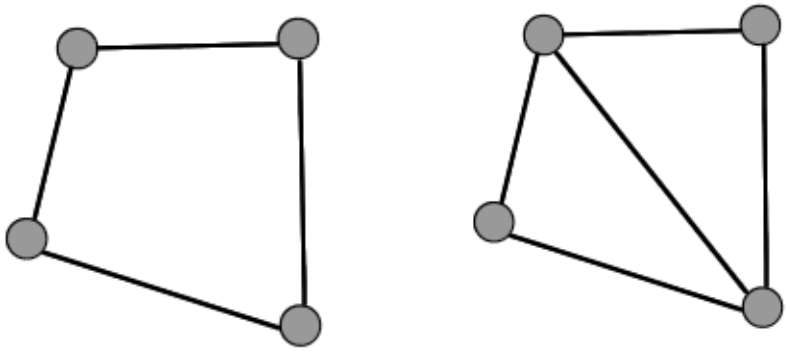
$$\dot{x}_i(t) = \sum_{j \epsilon N_i} \Big(\big(x_j(t) - y_j\big) - \big(x_i(t) - y_i\big) \Big) \tag{2}$$

Here, a set of points $y_1 ... y_n$ define the geometry of the desired formation, and points $x_1 ... x_n$ denote agent positions. Agent x_i is assigned to point y_i. As a result of this control law, the robots asymptotically approach a translated version of the desired formation regardless of the initial configuration. It can be shown that because this is a linear approach, the centroid of the final formation location matches that of the initial centroid of the agents. It should be noted that this approach requires all agents to have a common coordinate frame orientation.

Distance based formation control can be used to form translationally and rotationally invariant formations, i.e. formations that could take place with any rotation and any translation. Early work in this area was presented by Baillieul and Suri[3] and Olfati-Saber[17]. This approach works by having agents attempt to maintain a set of desired inter-agent distances, rather than inter-agent displacements. This is accomplished by using a weighted version of the consensus equation, where edge weights depend on the inter-agent distances, relative to their target distance as shown in equation 3. Here the formation is defined by a set of desired distances between neighboring agents, d_{ij}.

$$\dot{x}_i(t) = \sum_{j \epsilon N_i} \Big[\big(\|x_j(t) - x_i(t)\| - d_{ij}\big)\big(x_j(t) - x_i(t)\big) \Big] \tag{3}$$

Successful formation synthesis depends on two factors when using the distance



(a) An Example Flexible (Non-Rigid) Graph

(b) An Example Rigid Graph

Figure 2: Examples of Rigid and Non-Rigid Graphs. Nodes are represented by circles, edges are black lines.

based approach: the initial states of the agents and the graph geometry. Even if all agents are able to move such that they maintain the desired distances between their neighbors, it is not guaranteed that the formation has been successfully built. It is also necessary that the graph be *rigid*. Simply put, a graph is rigid if the structure formed by replacing the edges by rigid rods and the vertices by flexible hinges is rigid. If a graph is not rigid, it is said to be *flexible*. Figure 2 shows some examples of rigid and flexible graphs. Furthermore, even if a given graph is rigid, it is not guaranteed that the formation will be successfully built using the control law in equation 3. Agents may never be able to achieve their desired inter agent distances because it is possible that they get stuck in local minima. That is, the individual summation elements may be non-zero, but when added together they exactly cancel. Thus, successful formation synthesis also depends on the desired formation shape, and the agents' initial positions.

2.2.2 Distributed Assignment Algorithms

The second aspect of multi-agent formation control is determining what role each agent should have in the formation. In this context, assignment is defined as a one

to one mapping of agents to formation points, e.g. each agent must be mapped, or assigned, to a unique formation point. Thus, for a network of N agents there are $N!$ possible ways to assign the agents to roles in the formation. Furthermore, is it desirable to choose the assignment which maximizes some benefit or minimizes some cost. If we define a bijective function, $\pi[i]$, that maps agents to their assigned formation points, then the optimal assignment with respect to some cost function, C, satisfies:

$$\pi^* = \arg\min_{\pi} \sum_{i=1}^{N} C(\pi[i], x_i) \tag{4}$$

It is common to define the cost of assigning an agent to a formation point as the distance that agent must move to reach that point. It is highly desirable to reduce the complexity of assignment from $N!$ to polynomial time. The Hungarian Algorithm provides a centralized approach to the assignment problem and will yield the optimal assignment solution if the formation translation and rotation are known. This is discussed further in section 2.3. Finding an efficient decentralized solution to the assignment problem is a very difficult task.

One approach to this problem is to use market based algorithms [15] [25] [4], where robots bid on prospective assignments. Using this method, robots calculate a benefit of being assigned to each role. The robots then bid for potential assignments which dictate the cost of those assignments rending them more or less attractive for other robots to bid on. The approach proposed by Bertesekas [4] will find the optimal assignment, but it requires a central repository where the bids are stored, which essentially requires a complete communication graph topology. More distributed versions of auction algorithms are proposed by Michael [15] and Zavlanos [25] where no centralized bid repository is required and bidding is performed within each agent's neighbor set. However, in this distributed case an optimal assignment is not guaranteed. One clear requirement for use of these methods is the ability for robots to

communicate with other robots in their neighbor set. In both cases the formation pose must be known in advance for robots to be able to calculate the cost of assignment for a particular role. This also implies that all robots must have a common coordinate frame. It becomes a much more complex problem if robots must find an efficient solution without advance knowledge of the formation pose.

Another proposed solution to this problem is through the use of potential fields to guide robots to the nearest, unoccupied formation point [21] [23] [24]. Each formation point creates a potential well, which attracts nearby robots. When a robot knows an assignment point to be occupied, it removes its influence on the potential field. Robots identify occupied formation points either through local sensing [21] [23], or local communication [24]. The benefit of this approach is that it is truly distributed, and simultaneously solves assignment and formation synthesis. The downside to this approach is that agents must know the formation pose in advance, and hence have a common global coordinate frame. Furthermore, this approach is not very efficient as robots may have to travel to visit several possible destinations to discover if those roles are occupied.

2.3 The Hungarian Algorithm

The Hungarian algorithm was developed in 1955 by Khun [11] and in 1957 by Munkres [16] based on the work of two Hungarian mathematicians: Dnes Knig and Jen Egervry. The purpose of the algorithm is to reduce the complexity of finding the optimal assignment (from combinatorial to polynomial in time). A brute force approach to solving this problem would take $N!$ iterations, however the Hungarian algorithm can calculate the optimal assignment in $O(N^3)$.

The input to the algorithm is an $N \times N$ matrix where the element at the i^{th} row and the j^{th} column corresponds to the cost of assigning the i^{th} agent to the j^{th} role. One way to interpret the algorithm is by using a bipartite graph where there

are N vertices representing the agents, and N vertices representing the roles. Edges connect agents and roles, where each edge has a non-negative cost corresponding to the assignment cost. The Hungarian algorithm returns the set of edges that minimizes the sum of edge costs such that there is an edge connecting each node to a unique role.

2.4 The Iterative Closest Point Algorithm

The Iterative Closest Point Algorithm (ICP) is an algorithm commonly used in the field of computer perception to find the best match between two point clouds. It was introduced by Besl [5] and Chen [6] as a method for registration of 3D shapes. Given a set of points representing a model of an object, and a set of points representing a measurement of the object, the ICP algorithm attempts to find the translation and rotation of measurement points that best matches them with the object model. Often times the ICP algorithm is used to reconstruct 2D and 3D objects from multiple scans, or it is used to determine the 2D or 3D pose of an object given a measurement and a model of the object.

The ICP algorithm is initialized with a guess as to the pose (translation and rotation) of the object. The algorithm then consists of two basic steps: a matching phase, and a transformation phase. During the matching phase, each measurement data point is associated with its nearest model point. Then, given that association, during the transformation phase the optimal translation and rotation of the measurement data set is calculated such that it minimizes the mean square difference between measurement data points and their associated model points. Next, the measurement data is transformed according to the optimal translation and rotation. The algorithm then iterates back to the matching phase. This is repeated until convergence. The result is a translation and rotation that gives the best match between the two point clouds, for the given initial guess. It is not necessarily the globally optimal match

since the algorithm converges monotonically to the nearest local minimum [5].

The ICP algorithm is discussed because it was the inspiration for the algorithm presented in this thesis. Since the ICP algorithm attempts to find the best match between two point clouds, it seemed appropriate to apply it to the assignment problem where a set of robot positions (measurement data), must be matched with a formation model (model data). The only issue with applying the ICP algorithm directly is that it does not guarantee a one to one mapping of robots to role assignments because it uses nearest neighbor rules to associate points. This issue is resolved by applying the Hungarian algorithm to perform the association, rather than using the nearest neighbor approach. This simple change resulted in the development of the assignment and formation control algorithm which is presented in the following chapters.

CHAPTER III

THEORY

This chapter introduces the assignment algorithm in detail and derives its stability and convergence properties. Methods for implementation on a network of mobile robots are given. This is first done for the complete graph case where each robot has complete information on the relative positions of all other robots. That is followed by a discussion of the case where robots only have access to local information. Next, a slight variation of the assignment algorithm is introduced where one robot is designated the leader with a fixed assignment. Lastly, a comparison is made with previous methods of formation control with a discussion of the advantages and disadvanges of the proposed assignment algorithm.

3.1 Problem Definition

The aim is to develop a decentralized algorithm for each agent in a network to identify the pose (translation and rotation) of a formation and to correctly assign itself to a unique position in the formation. Each agent will move towards its self-assigned position, and as time goes to infinity, the positions of the agents should exactly match those of a rotated and translated version of the desired formation.

3.2 The Assignment Algorithm

Inspired by the Iterative Closest Point (ICP) algorithm, an algorithm has been developed based on sum of squares minimization techniques. Using neighbor displacement information, each agent tries to identify the optimal pose (translation and rotation) of the formation model. Then, based on this rotation and translation of the model, each agent assigns itself and its neighbors to points in the formation. Nodes then

move towards their target positions. This process is performed every time step, thus agents are dynamically updating their pose estimates as well as their assignments as they are moving. The algorithm presented here applies to 2 dimensional space, however this method could be applied to an arbitrary number of dimensions.

3.2.1 Definition of the Assignment Algorithm

Consider a network of N mobile agents where $X = x_1, x_2, ..x_N \in \mathbb{R}^2$ denotes the position of each agent. The agents begin with some arbitrary initial configuration, X_0, and we would like them to form some translationally and rotationally invariant formation. The desired formation model is defined by: $P = p_1, p_2...p_n \epsilon \mathbb{R}^2$. These points should be chosen such that their centroid is at the origin: $0 = \sum_{i=1}^{N} p_i$. Let the assignment set Y be any bijection of P such that agent x_i is assigned to element y_i. Furthermore, let $\tau \epsilon \mathbb{R}^2$ denote the translation of the formation and θ denote the rotation of the formation. We would like to select Y, τ, and θ as to minimize the cost function defined by equation 5. A solution is said to be optimal if it is the minimizer of the given cost function.

$$L(Y, \tau, \theta) = \sum_{i=1}^{N} ||R(\theta)y_i + \tau - x_i||^2 \tag{5}$$

Where $R(\theta)$ is the rotation matrix:

$$R(\theta) = \begin{pmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{pmatrix} \tag{6}$$

Assignment Algorithm

1. Choose an initial guess as to the rotation, θ, and translation, τ, of the formation: $\tau[0] = \tau_0$, $\theta[0] = \theta_0$.

2. Increment iteration variable, k. Given the translation and rotation from the

previous step, choose the assignment set, Y, that minimizes the cost function:

$$Y[k] = \arg\min_{Y} L(Y, \tau[k-1], \theta[k-1]) \tag{7}$$

This can be done using existing methods such as the Hungarian algorithm.

3. Given the assignment determined in step 2, calculate the optimal rotation angle, θ, and translation vector, τ, that minimizes the cost function:

$$(\tau[k], \theta[k]) = \arg\min_{[\tau,\theta]} L(Y[k], \tau, \theta) \tag{8}$$

It can be shown that the optimal rotation angle, θ, is given by:

$$\theta = tan^{-1}\left(\frac{W_2}{W_1}\right) \tag{9}$$

$$W_1 = \sum_{i=1}^{N} (x_i - \mu_x)^T (y_i - \mu_y) \tag{10}$$

$$W_2 = \sum_{i=1}^{N} (x_i - \mu_x)^T \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} (y_i - \mu_y) \tag{11}$$

where μ_x and μ_y are the respective centroids of sets x and y. The optimal translation vector, τ, is determined by aligning the centroids of the two point sets:

$$\tau = \frac{1}{N}\sum_{i=1}^{N} x_i - \frac{1}{N}\sum_{i=1}^{N} y_i \tag{12}$$

If the algorithm has not converged, jump back to step 2 with the new estimates for τ and θ.

3.2.2 Convergence Proof

3.2.2.1 Theorem 1

The assignment algorithm converges in finite time.

Proof. Let us examine the value of the cost function following each step of the algorithm, at iteration k. The error following assignment in step 2 is given by:

$$e[k] = \min_{Y} L(Y, \tau[k-1], \theta[k-1]) \tag{13}$$

Then in step 3, following translation and rotation, the error can be written as:

$$d[k] = \min_{\tau,\theta} L(Y[k], \tau, \theta) \tag{14}$$

Note that $d[k] \leq e[k]$. This is because both $e[k]$ and $d[k]$ are the values of the cost function for some fixed assignment $Y[k]$. However, $d[k]$ is the cost after optimal translation and rotation of the formation given $Y[k]$. If $d(k) > e(k)$ then the rotation and translation calculated in step 3 is suboptimal. Then, the algorithm jumps back to step 2 and a new optimal assignment is calculated for the given translation. Because we are not changing τ and θ, $e[k+1] \leq d[k]$. Again, if $e[k+1] > d[k]$ then the new assignment is suboptimal. Thus:

$$0 \leq d(k+1) \leq e(k+1) \leq d(k) \leq e(k) \tag{15}$$

Finally, since there are a finite number of sets Y that are bijective with set P, the algorithm converges in finite time. □

3.2.2.2 Corollary to Theorem 1

The assignment algorithm converges to a Nash equilibrium. In the sense of game theory, the two "players" consist of the formation pose update (step 3), and the role assignment update (step 2). Both players attempt to minimize some utility function, which in this case is the cost function given by 5. Thus, convergence to a Nash equilibrium means the final assignment is optimal given the final pose, and the final pose is optimal given the final assignment:

$$Y^* = \arg\min_{Y} L(Y, \tau^*, \theta^*) \tag{16}$$

$$(\tau^*, \theta^*) = \arg\min_{\tau,\theta} L(Y^*, \tau, \theta) \tag{17}$$

Proof. When the algorithm converges at iteration k_f, we have:

$$Y[k_f] = Y[k_f - 1] = Y^* \tag{18}$$

$$(\tau[k_f], \theta[k_f]) = (\tau[k_f - 1], \theta[k_f - 1]) = (\tau^*, \theta^*) \tag{19}$$

Inserting (18) and (19) into (7) and (8) yields:

$$Y^* = \arg\min_{Y} L(Y, \tau^*, \theta^*) \tag{20}$$

$$(\tau^*, \theta^*) = \arg\min_{\tau,\theta} L(Y^*, \tau, \theta) \tag{21}$$

□

3.2.2.3 Complexity

Due to the combinatorial nature of the assignment problem, along with the geometric dependence of the algorithm, it is difficult to derive an upper bound for the complexity of the assignment algorithm. We know there is an absolute upper bound of $N!$ possible iterations since that is the maximum number of permutations of Y, however empirical testing shows the algorithm converges in just a few iterations for $N = 1...50$. Furthermore, we know most of the $N!$ possible permutations of Y will never occur because they will always be suboptimal solutions to the assignment calculated in step 1 of the assignment algorithm. To get a better upper bound estimate, we have attempted to find the maximum number of possible assignments that can occur during execution of the algorithm. From (12) we know the centroid of the formation will always be aligned with the centroid of the agents, regardless of the assignment Y. Thus we can hold τ constant, and vary θ from 0 to 2π and count the number of unique solutions to the minimization problem in step 1 of the assignment algorithm. This will give a rough estimate on the maximum number of states, and therefore give a rough estimate as to the upper bound of the complexity of the assignment algorithm.

To this end, we have used simulation to find the number of possible assignments vs. network size, N. We tested values of N ranging from 1 to 56 in increments of 5

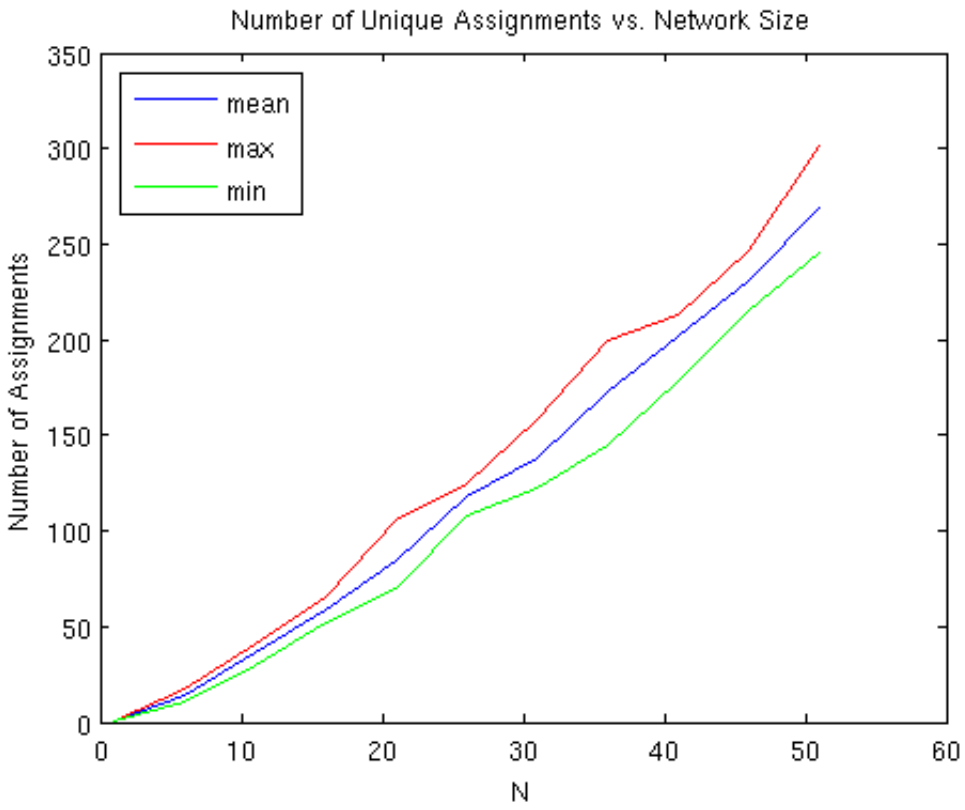


Figure 3: Number of Unique Assignments vs. Network Size, based on simulation with random data sets

($N = 1, 6, 11...56$). For each N, the X, Y coordinates of agent positions and formation points were randomly generated over an interval of 0 to 1. A brute force approach was used to finely sample assignments for values of θ ranging from 0 to 2π. This was repeated 10 times for each value of N, each time with different random formations and agent distributions. Based on these 10 trials, the mean, min and max number of unique assignments were computed for each value of N, see Fig.3 .

This plot suggests a nearly linear relationship between network size and the number of possible assignments. Based on this empirical data, we estimate the assignment algorithm to terminate in at most $O(N)$ iterations. Since assignment in step 1 runs in $O(N^3)$, and translation/rotation step 2 runs in $O(N)$, we can estimate the complexity for the overall algorithm to be $O(N^4)$. In practice however, the assignment algorithm converges much faster, and in most cases convergence is achieved in just a

few iterations (2-5 iterations for $N = 1..56$).

3.2.3 Distributed Implementation on Mobile Robot Systems

Decentralized assignment and formation control of a network of mobile robots is one application of the proposed algorithm. In this scenario, each robot can sense the relative displacements of all other robots in the network, however each robot may use its own coordinate system. Each robot uses the assignment algorithm to calculate its own assignment as well as the pose of the formation. The result gives the robot a relative point in space to move towards. Then, while in motion, each robot will continuously run the assignment algorithm to re-evaluate its assignment and the formation pose. This adds robustness to the system, and makes localization unnecessary. Furthermore, this allows for dynamic assignment where agents can alter their assignments for changing conditions.

Initially at time, t=0 all agents will independently arrive at the same assignment, and formation pose if at least one of the two following statements is true:

1. All agents use the same set of initial guesses (formation rotation and translation). If all agents use the same initial guesses, then they will invariably converge to the same assignment and formation pose. One approach could be to have each robot use the moment alignment method for efficient assignment as outlined in section 3.2.3.1. This way each robot will converge to the same result with just two initial guesses.

2. Each agent uses a sufficient number of initial guesses as to converge to the optimal assignment with optimal rotation and translation of the formation. Since all agents have the same neighbor set and model set, and there is a unique optimal solution, then all agents would converge to the same solution.

After the first execution of the algorithm, each agent will use its previous result its next initial guess provided the cost function continues to be reduced with time.

However, if the cost function actually increases at any point in time, a more complete set of initial guesses should be used. Thus we assume all agents converge to the same assignment and formation pose at time, t: $Y[t]$, $\tau[t]$, and $\theta[t]$. Each agent will use the following control law to reduce the cost function at each instance in time:

$$x_i[t+1] = x_i[t] + \delta\Big(R(\theta[t])y_i[t] + \tau[t] - x_i[t]\Big), \quad 0 < \delta < 1 \tag{22}$$

The error of that particular assignment and formation pose at time $t+1$ becomes:

$$\begin{aligned} L[t+1] &= (1-\delta)^2 \sum_{i=1}^{N} \Big|\Big| R(\theta[t])y_i + \tau[t] - x_i[t] \Big|\Big|^2 \\ L[t+1] &= (1-\delta)^2 L[t] \end{aligned} \tag{23}$$

Thus the error of a particular assignment and formation pose decreases with every time step. Each agent will use its previous result as its next initial guess so we are guaranteed $L[t+1] \leq (1-\delta)^2 L[t]$. So the cost function decreases asymptotically to zero as t goes to infinity. If the cost function equals zero, the agents must then be in the desired formation (zero distance between each agent and its assigned point). These results are verified in the simulation chapter, and are further demonstrated by experiments described in the experimentation chapter.

3.2.3.1 Methods for Efficient Matching

Note that the assignment algorithm does not necessarily converge to the optimal solution that satisfies $\min_{Y,\tau,\theta} L(Y,\tau,\theta)$. Rather, it will converge to one of many Nash equilibria. Which equilibrium it converges to is entirely dependent on the initial guess used. One can increase the probability of finding the optimal solution by executing the algorithm multiple times, each time with a different initial guess. From (12), the optimal translation is given by aligning the centroids of the point sets, regardless of assignment. Therefore, a good initial guess for τ would equal the difference between the centroids of the formation points and agent locations. One could hold the initial

guess for τ constant while varying the initial guess for θ. A brute force approach using a sufficient number of evenly spaced angular guesses around the unit circle would yield the optimal solution. The number of guesses required for optimal assignment depends both on network size, N, and complexity of the desired formation. The number of initial guesses needed could be determined experimentally through exhaustive off line testing.

Under certain conditions, one can achieve a high probability of finding the optimal assignment with just two initial guesses. This approach was proposed by Besl for use in optimal matching of the ICP algorithm [5], and it works equally well for the purposes of the assignment algorithm. First, choose τ as to align the centroids of the model points and agent locations as mentioned above:

$$\tau = \mu_x - \mu_y \tag{24}$$

$$\mu_x = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{25}$$

$$\mu_y = \frac{1}{N}\sum_{i=1}^{N} y_i \tag{26}$$

Next, let us define the moments of distribution geometry as:

$$M_x = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu_x)(x_i - \mu_x)^T \tag{27}$$

$$M_y = \frac{1}{N}\sum_{i=1}^{N}(y_i - \mu_y)(y_i - \mu_y)^T \tag{28}$$

The principle moments of distribution are defined by the eigenvectors of the moment matrices. Since we are operating in a plane, the moment matrix will be 2×2 yielding two orthogonal principle moment vectors. Let $U_1, U_2 \epsilon \mathbb{R}^2$ be the eigenvectors of M_x and $V_1, V_2 \epsilon \mathbb{R}^2$ be the eigenvectors of M_y. We can check to see how distinct these principle moments are by comparing their respective eigenvalues. Let λ_1 and λ_2 be the eigenvalues of M_x such that $\lambda_1 > \lambda_2$, and let ν_1 and ν_2 be the eigenvalues of M_y such that $\nu_1 > \nu_2$. Let us define principle moment distinctness, α, by:

$$\alpha_x = \frac{\lambda_2}{\lambda_1} \tag{29}$$

$$\alpha_y = \frac{\nu_2}{\nu_1} \tag{30}$$

If α_x and α_y are sufficiently small, e.g. $\alpha_x, \alpha_y < .7$, then the principle moments are sufficiently distinct, and one can reliably use just two initial guesses to find an efficient solution to the assignment problem. This is done by rotating the formation model such that the principle moments U_1 and V_1 are aligned. This requires two guesses to check both the parallel and anti-parallel cases. Thus, the two initial guesses for rotation would be:

$$\theta_1 = \quad \angle U_1 - \angle V_1 \tag{31}$$

$$\theta_2 = \quad \angle U_1 - \angle V_1 + \pi \tag{32}$$

Here the $\angle$ operator denotes taking the angle of a vector.

3.2.3.2 Robot Trajectories

Following the above procedure, we would expect all robots to initially converge to the same result. Then, assuming the robots are holonomic, they would then begin moving in straight lines to their assigned points. Following Bellman's principle of optimality, if agents act optimally from the start, then the optimal solution does not change. Thus, the robots would continue to move in straight lines until they reach the destinations they chose initially.

Furthermore, the optimal assignment will rarely result in any two robot trajectories crossing. To gain some insight into this phenomenon, let us consider a hypothetical case with two robots where we use a linear cost function instead of a quadratic one. In this case, there are two possible assignment solutions: one where the trajectories cross, and one where they do not. Figure 4 shows the distances the robots must

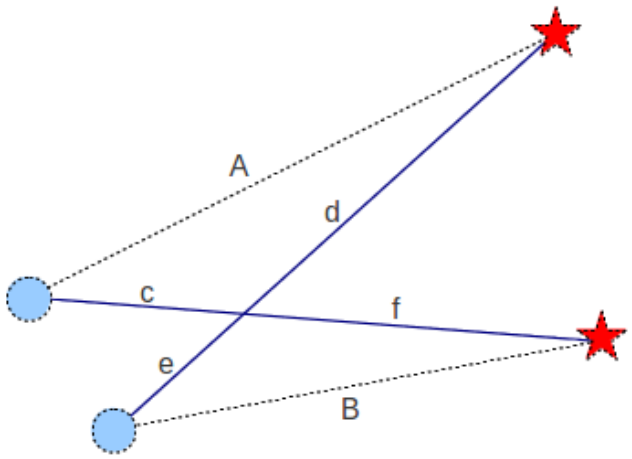


Figure 4: Scenario with assignment of two robots. Robots are represented by two circles, and destination points shown by stars. Two possible assignments: Solid lines show assignment with crossing paths, dotted lines show assignment with no crossing paths. Letters denote the lengths of line segments.

travel according to their assignments. Since we are using a linear cost, the cost for the assignment with no crossing paths is $L_1 = A + B$ and the cost of the assignment with crossing paths is $L_2 = c + d + e + f$. Note that $A < c + d$ and $B < e + f$, thus $L_1 < L_2$. Since the Hungarian Algorithm will always choose the assignment with the minimal cost, this shows that no two robot trajectories will intersect if a linear cost function is used.

However for the purposes of this algorithm, we are using a quadratic cost function so the above analysis will not hold. There are some scenarios where the optimal assignment would involve crossing trajectories. However, in the vast majority of cases, even with a quadratic cost function, the robot trajectories do not intersect for the same reasons outlined above.

The reason we are concerned with crossing trajectories has to do with inter-robot collisions. If robot trajectories do not intersect, then robots will inherently not collide with each other en route to building the formation. As a result of this behavior, it is much easier to avoid inter-robot collisions when dealing with a large number agents.

This is in contrast to fixed-assignment formation control methods where collision avoidance while building the formation could be a major issue.

3.3 Variations on the Algorithm

Initially, the complete graph case was used for its ease of development and analysis. However, it would be highly desirable to know the properties of the algorithm for various scenarios. This section discusses two cases where the algorithm must be modified to meet different needs. The first is the case where robots only have access to local information. That is, the positions of far-away robots are unknown. The second is the case where we would like to have one robot be designated the leader. This scenario is useful for leader-follower type formations where you would like to have a single leader dictate the position of the formation.

3.3.1 The Non-Complete Graph Case

The first obvious difference between the complete graph case and the non-complete graph case is that there will be fewer observable robots than formation points. However the ultimate goal is still the same: determine the formation pose and assignment that minimize the cost function for the observable robots. While the algorithm itself will still operate and converge with this inequality, the previous methods used to determine the initial guess are no longer valid. Previously, the optimal translation was independent of the optimal assignment (the optimal translation was always to align the centroids of the robot distribution and model distribution). This made it very easy to choose the initial guess for translation. Now, the initial translation guess is no longer obvious because each node does not know the centroid of the network, it knows only the centroid of its neighbor set. Furthermore, the final result will be highly dependent on the initial translation guess because this will determine in which region of the formation model robots are initially assigned. Thus to find the optimal solution, or even an efficient one, multiple translation initial guesses must be used in

addition to multiple rotation initial guesses.

3.3.1.1 Approach

One approach to solving the problem of multiple initial translation and rotation guesses is to treat the formation model as a collection of smaller formation models. Say there are N_i robots in robot i's neighbor set, and there are M points in the formation model. A brute force method would be to choose N_i points out of the M model points and run the assignment algorithm as if it were the complete graph case. Then, choose a different set of N_i points from M and run the algorithm again. Repeat this process for all the possible ways of choosing N_i points out of set M. This would require running the assignment algorithm $\binom{M}{N_i}$ times! However, if something is known about the structure of the graph this number can be greatly reduced. For example, if the graph was known to be a δ-disk proximity graph, then only reasonable sets of N_i must be tried. Only points from the model M within a δ radius of each other need to be used.

3.3.1.2 Analysis

Analysis of the non-complete graph case has proven to be much more difficult than the complete graph case. Because the algorithm is inherently non-linear, the usual tools for the analysis of network control systems are not applicable. Previously, it was assumed that robots would independently converge to the same result because each robot was using the same information. This can no longer be assumed, however, because now each robot can only observe a portion of the overall network. Given this uncertainty, there are two conditions that must hold for formation synthesis to be successful in the non-complete graph case: assignment consistency, and system stability.

First, all robots must converge to a consistent role assignment. In other words, each robot must assign itself to a unique role in the formation. If each robot is

able to find the optimal assignment for its given neighbor set then this assignment must agree with every other robot's assignment choice. It is not obvious under what circumstances this will be the case. Surely if the robots start out in the proper formation, then the optimal assignment cost is zero. If every robot converges to a result with zero cost, then all robots have converged to a consistent assignment. This thought can be extended to the case where robots don't start out exactly in the proper formation, but are perturbed by some very small amount. Then if each robot converges to the optimal result for their neighbor set, the result should still be consistent. The question then becomes, how much can the robots be perturbed from the proper formation before the assignments are no longer consistent? At this point we do not have an answer to this question, but intuitively it should somehow depend on the density of the network. If we have almost a complete graph, then the robots could start almost randomly and still converge to a consistent result. However, if we have a very sparse graph then the robots must start very close to the desired formation. These hypotheses were tested via simulation, and a review of the results are posted in the simulation chapter.

The second condition necessary for successful formation synthesis is that, given the role assignment, the control law must be stable. As stated above, each agent will use the following control law:

$$x_i[t+1] = x_i[t] + \delta\Big(R(\theta[t])y_i[t] + \tau[t] - x_i[t]\Big), \quad 0 < \delta < 1 \tag{33}$$

The stability proof for the complete graph case relies upon the fact that each agent will agree on the formation translation and rotation. In the non-complete graph case, even if agents agree upon the assignment as described above, they may not immediately agree upon the formation translation and rotation. Since this is a non-linear control law, some type of Lyapunov stability analysis is required. We have so far been unable to identify a proper Lyapunov function that demonstrates the system stability, although numerous simulations suggest this control law yields a globally

stable system for non-complete graphs. These simulation results are discussed in more detail in the simulation chapter.

3.3.2 Leader Based Assignment and Formation Control

Often times in formation control scenarios, it is desirable to have a single robot be designated as the "leader". This leader robot does not run the algorithm as everyone else, but acts on its own. The leader could be driven by some other goal, such as navigate to a goal, or it could even be controlled directly by a human. The leader doesn't necessarily have to be a robot at all; it could be any object of interest. However, the other robots still attempt to build a formation that includes the leader. The result is having the formation moved along with the leader. This method is particularly useful if you want to move a group of robots while in formation, such as a convoy, or Unmanned Aerial Vehicle (UAV) formations. As mentioned in the introduction, if you want to move a group of robots from point A to point B, only the leader needs to know how to get there and the rest of the robots follow along simply by maintaining the formation. This is also useful in protection or escort type scenarios where you would want the robots to maintain some formation around a target.

3.3.2.1 Approach

Here we return our analysis to the complete graph case, where every robot knows the instantaneous position of every other robot. We can actually achieve the behavior described above without any modifications to the algorithm. If one robot did not move according to the algorithm, the formation pose would slowly be pulled to include that robot. However, this is not ideal for two reasons. First, it is inefficient. The assignments and pose calculations are done assuming that all robots will be moving to their assigned points. Initially, all follower robots will begin driving towards the points using this assumption. When the leader doesn't act as expected, the followers

then have to modify their targets accordingly. The result is having robots drive extra distance unnecessarily. This scenario is demonstrated in the simulation chapter. The second problem is that it is slow to converge. If the robots are in formation, then the leader begins to drive, the followers will lag significantly behind. This is because the translation of the formation is dictated by the centroid of all the robots. If there is a network of 4 robots, then the centroid is moved by $\frac{1}{4}$ the distance traveled by the leader. If there are 100 robots, the centroid is only moved by $\frac{1}{100}$ the distance traveled by the leader.

Both of these issues can be readily solved with a slight modification to the assignment algorithm. These modifications require the follower robots to be able to identify who the leader is. First, during the assignment phase, the assignment of the leader should be fixed. This can easily be done when running the Hungarian Algorithm by setting the cost of assigning the leader to zero for that particular role. Then, make it very costly to assign the leader to any other role in the formation. Next, during the pose update phase, we will constrain the formation pose such that the leader defines where its point should be. Rather than translating the formation so that the centroids are aligned, translate the formation so that the leader's position is aligned with its assigned formation point as shown in equation 34. Next, rather than rotating the formation about the centroid, rotate it about the leader's position as shown in equations 35 to 37.

$$\tau = x_{leader} - y_{leader} \tag{34}$$

$$\theta = tan^{-1}\left(\frac{W_2}{W_1}\right) \tag{35}$$

$$W_1 = \sum_{i=1}^{N}(x_i - x_{leader})^T(y_i - y_{leader}) \tag{36}$$

$$W_2 = \sum_{i=1}^{N} (x_i - x_{leader})^T \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} (y_i - y_{leader}) \tag{37}$$

Here x_{leader} is the leaders position and y_{leader} is the leader's assigned formation point. Let us redefine the cost we are trying to minimize to reflect the fact that we are constraining the position of the formation about the leader:

$$L(Y, \theta) = \sum_{i=1}^{N} ||R(\theta)(y_i - y_{leader}) + \tau + (x_i - y_{leader})||^2 \tag{38}$$

Notice we are no longer minimizing over τ because it has been constrained such that the leader's position is aligned with its assigned formation point.

3.3.2.2 Analysis

The problems of inefficient trajectories and slow convergence have been eliminated. The formation position now follows the leader exactly, so follower robots will move directly towards the appropriate points. If the leader begins to move, the formation moves in exact alignment with the leader. The convergence properties of the algorithm remain unchanged. These results are verified in the simulation chapter, and are further demonstrated by experiments described in the experimentation chapter.

3.4 Comparisons With Previous Work

This approach to assignment and formation control is unique when compared to most previous work in this area. Using our approach, robots simultaneously identify the formation pose in addition to role assignment. This is sufficient information to give each robot a target position in space. Thus, both role assignment and formation synthesis are accomplished simultaneously. This is in contrast to other approaches to building rotationally and translationally invariant formations where assignment is completed before the formation building stage takes place. This is the most fundamental difference between our approach and previous work. Further differences, advantages and

disadvantages are discussed in detail in the sections below. Attributes are broken down into three categories: required information at the robot level, efficiency, and robustness.

3.4.1 Required Information at the Robot Level

As derived above, the pieces of information required at the robot level for our assignment algorithm to be guaranteed to converge are: the relative positions of all robots, and knowledge of the desired formation model. Note that the relative robot positions can be given in an arbitrary coordinate system, so no global coordinate system is necessary and each robot may have its own coordinate frame. This differs from displacement based formation control methods where robot coordinate frames must have some common orientation. This also differs from the potential field based assignment protocol proposed by Zavlanos and Pappas [21] where robots use potential fields to move into formations. It also is necessary for most market based distributed assignment protocols for robots to have a common coordinate frame[15]. Furthermore, it is not necessary for robots to discern the identity of the other robots using our algorithm. This is in contrast to displacement based and distance based methods of formation control where the identity of neighbors must be known. Additionally, distance based or displacement based formation control methods require the role assignment of robots to be given in advance.

The price for needing only robot displacement information is that each robot needs *complete* displacement information for the entire network. This is the greatest downside to this approach to formation control. This means that it is not a truly distributed method for formation control. There are circumstances where complete information is not required, but as discussed above, we are unable to prove convergence in this case. Much previous work on formation control, including displacement and distance based approaches, require only local position information. Thus, for

large swarms of robots where it would be impossible to identify the positions of all agents, our assignment algorithm is not the right approach.

3.4.2 Efficiency

A great advantage of our algorithm over previous methods is its efficiency with respect to the distances robots have to drive to build the formation. The goal of the algorithm is to find the most efficient formation pose and role assignment given the state of the system, and if a sufficient number of initial guesses are used, the optimal solution can be found. It can be shown that the displacement based formation control methods result in using the optimal translation of the formation, but they are not necessarily efficient when it comes to formation rotation and assignment.

The efficiency of other assignment methods such as the Hungarian Algorithm, or market based methods, is reduced because the formation pose is often fixed in space. The Hungarian Algorithm alone returns the optimal assignment for a given formation pose, but does nothing to find an efficient location for the formation to take place. Similarly, approaches used in [15] and [21] also assume a fixed, or previously agreed upon formation pose, which may be inefficient.

With respect to computational efficiency, this algorithm is lacking. Consensus based formation control algorithms require very little computational power when compared to this assignment algorithm. As described above, we estimate the complexity of our approach to be $O(N^4)$, where as the complexity of consensus based protocols is $O(N)$.

3.4.3 Robustness

Our algorithm has two advantages over the distance based and displacement based formation control methods with respect to robustness. First, robots cannot get stuck in "local minima", which can happen in the distance based formation control method. It is also globally asymptotically stable when you have a complete graph, something

Table 1: Comparison of Formation Control Methods

Method	Graph Type Required	Requires Robot Identification	Requires Common Orientation	Local Minima	Fixed Assignment	Complexity
Distance Based	Rigid	Yes	No	Yes	Yes	$O(n)$
Displacement Based	Connected	Yes	Yes	No	Yes	$O(n)$
Assignment Algorithm	Complete	No	No	No	No	$O(n^4)$

that cannot be said for the distance based formation control method. Second, as mentioned previously, robot trajectories rarely intersect as robots move into their positions. This means there are fewer possibilities for collisions and robots do not need to deviate from their trajectories to avoid collisions. Collision avoidance and the resulting trajectory deviation is a problem not often considered in the analysis of other formation control techniques.

3.4.4 Summary

In summary, our algorithm offers some advantages in efficiency and robustness over previous methods of formation control. The greatest downside to our approach is that a complete graph is required for guaranteed stability, and the algorithm therefor cannot be considered to be a completely distributed one. The other main downside to our algorithm is the computation complexity when compared with consensus based methods. These comparisons are summarized in table 1.

CHAPTER IV

SIMULATION

To verify and supplement our theory, we implemented the proposed assignment algorithm in Matlab and simulated it using a virtual network of robots. The robots were modeled as points in space with zero inertia. The output of the algorithm was a velocity command for each robot at each point in time. The robots can instantaneously move in any direction (i.e. they are holonomic). In all of the following scenarios, each robot uses its own coordinate frame and does not know the identity of its neighbors unless otherwise noted.

4.1 Complete Graph Case Simulations

First, we implemented the algorithm on a network of robots with a complete graph where each robot knows the relative positions of all other robots. The initial positions of the robots were randomly generated within a 4×4 square. Three tests are shown for networks of varying size: $N = 8, 15, 29$ with all robots running the same algorithm. Figs. 5-7 show the robot initial and final positions as well as paths taken. The algorithm scaled quite easily from $N = 8$ to $N = 29$, and did not require any modification to do so. Notice that all robots travel in straight lines to unique points in the desired formation. This shows that the assignment and formation pose do not change as each robot moves towards its assigned point. By inspection we can see that no robot has to travel exceedingly far to reach its destination, as we would expect. Furthermore, we can observe that the centroid of the final configuration matches the centroid of the initial configuration. It should also be noted that in all three of these instances the robot trajectories do not intersect, even with 29 robots. The results of these simulations match theoretical properties very closely, and formation synthesis

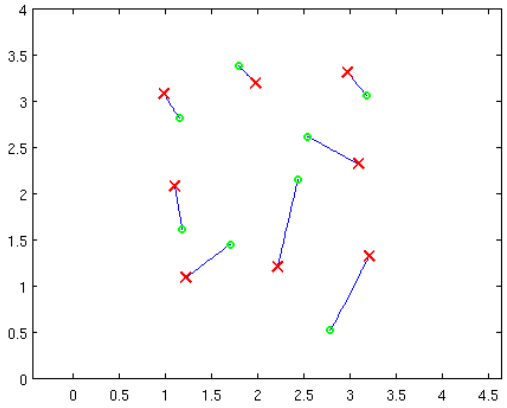


Figure 5: Simulation data for a network of 8 robots moving to a box formation. Circles show the initial states, Xs show the final states, and lines show the paths taken.

is always successfully achieved in the complete graph case.

Fig. 8 repeats the N=8 scenario, however we simulate a malfunctioning robot by holding its location fixed. Notice the curved lines, indicating the formation pose seen by each robot is changing with time in reaction to one robot not moving. This occurs as a result of each robot re-evaluating the formation pose and assignment at each time step. Not evident in the figures, is that convergence time is greatly increased if one robot remains stationary. This is because the formation pose is very slowly pulled towards the stationary robot as discussed in section 3.3.2.

To further demonstrate the dynamic nature of the algorithm, and reaction to disturbances, we fixed the assignment of one robot. Fig 9 shows the case where a single robot has a fixed assignment that is unknown to the other robots. As the robot with fixed assignment behaves unexpectedly, the other robots must modify their paths to compensate. Notice the sharp changes to the paths taken by 3 of the robots. This shows those robots dynamically changing their assignments in reaction to the pre-assigned robot's motion.

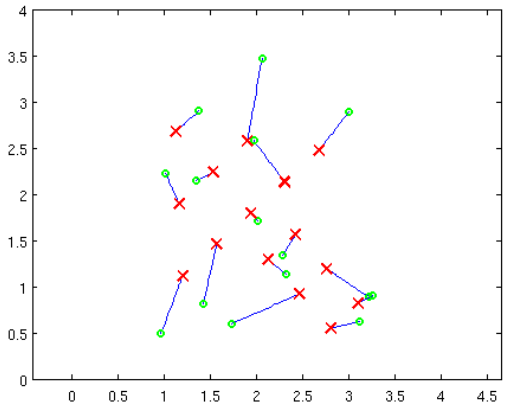


Figure 6: Simulation data for a network of 15 robots moving to an arrow formation. Circles show the initial states, Xs show the final states, and lines show the paths taken.

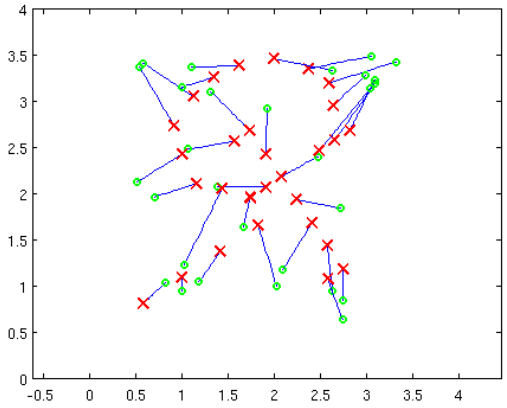


Figure 7: Simulation data for a network of 29 robots moving to a "GT" formation. Circles show the initial states, Xs show the final states, and lines show the paths taken.

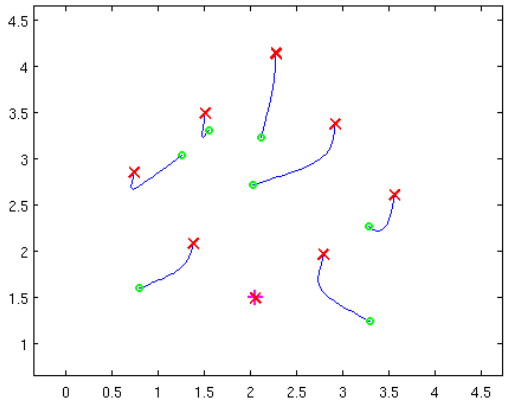


Figure 8: Simulation data for a network of 8 robots moving to a box formation. The '*' robot in the bottom center does not move. Other robots dynamically change the pose of the formation to compensate.

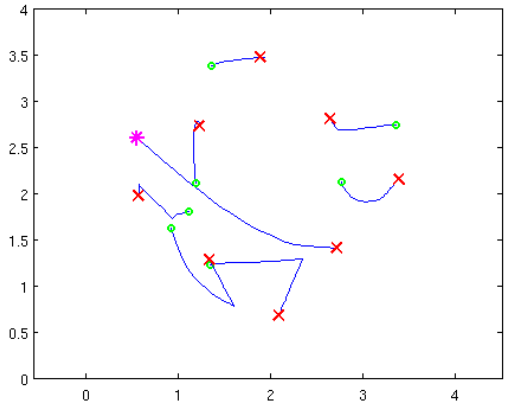


Figure 9: Simulation data for a network of 8 robots moving to a box formation. The '*' robot has a fixed assignment to a point on the edge. Other robots dynamically change pose and assignment to compensate.

4.1.1 Number of Nash Equilibria

As discussed in the Theory chapter, the assignment algorithm will converge to a Nash Equilibrium where the final assignment is optimal given the final pose, and the final pose is optimal given the final assignment. Given a particular formation model, and robot distribution, there exist several Nash Equilibria. Which equilibrium point is reached depends entirely on the initial guess used. In section 3.2.2.3 we examined the number of optimal assignments given that the centroid of the robot positions and the centroid of the formation model are aligned. This gave us an upper bound for the number of possible assignments the algorithm could visit on its way to convergence. To further expand upon this, we wanted to find the number of Nash Equilibria.

We expect the set of Nash Equilibria to be a subset of the total number of possible optimal assignments. To find the approximate number of Nash Equilibria vs network size, we tested values of N ranging from 1 to 56 in increments of 5 ($N = 1, 6, 11...56$). For each N, the x, y coordinates of agent positions and formation points were randomly generated over an interval of 0 to 1. A brute force approach was used to finely sample initial guesses for values of θ ranging from 0 to 2π. The assignment algorithm was executed for each initial guess, and the number of unique results was tallied. This was repeated 10 times for each value of N, each time with different random formations and agent distributions. Based on these 10 trials, the mean, min and max number of unique Nash Equilibria were computed for each value of N. This is shown in Fig.10 along with the total number of optimal assignments.

This shows that the algorithm is successful in eliminating many optimal assignments that are not also Nash Equilibria. It is interesting to note that the number of Nash Equilibria scales roughly 1:1 with the number of nodes in the network.

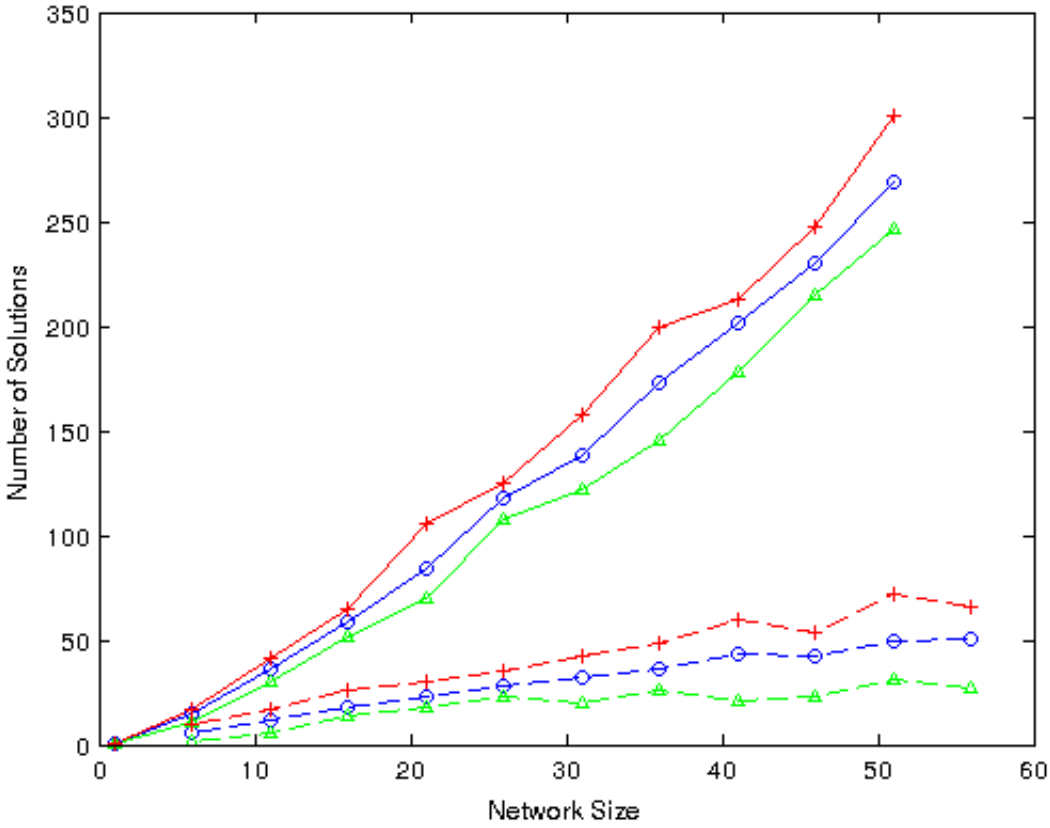


Figure 10: Simulation to compute the number of solutions vs. network size. 10 trials for each network size, N. Given that the centroids are aligned, the solid lines show the total number of optimal assignments. Number of Nash Equilibria shown with dotted lines. Means shown with circles. Maxes shown with +'s. Mins shown with triangles.

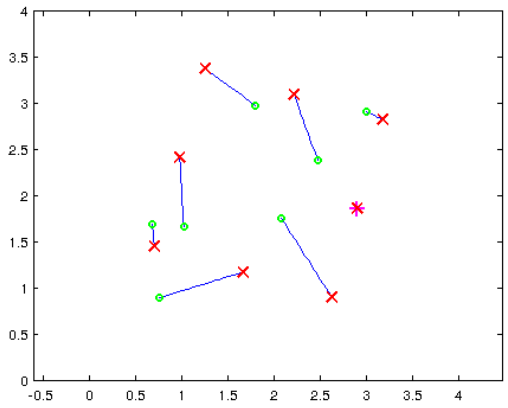


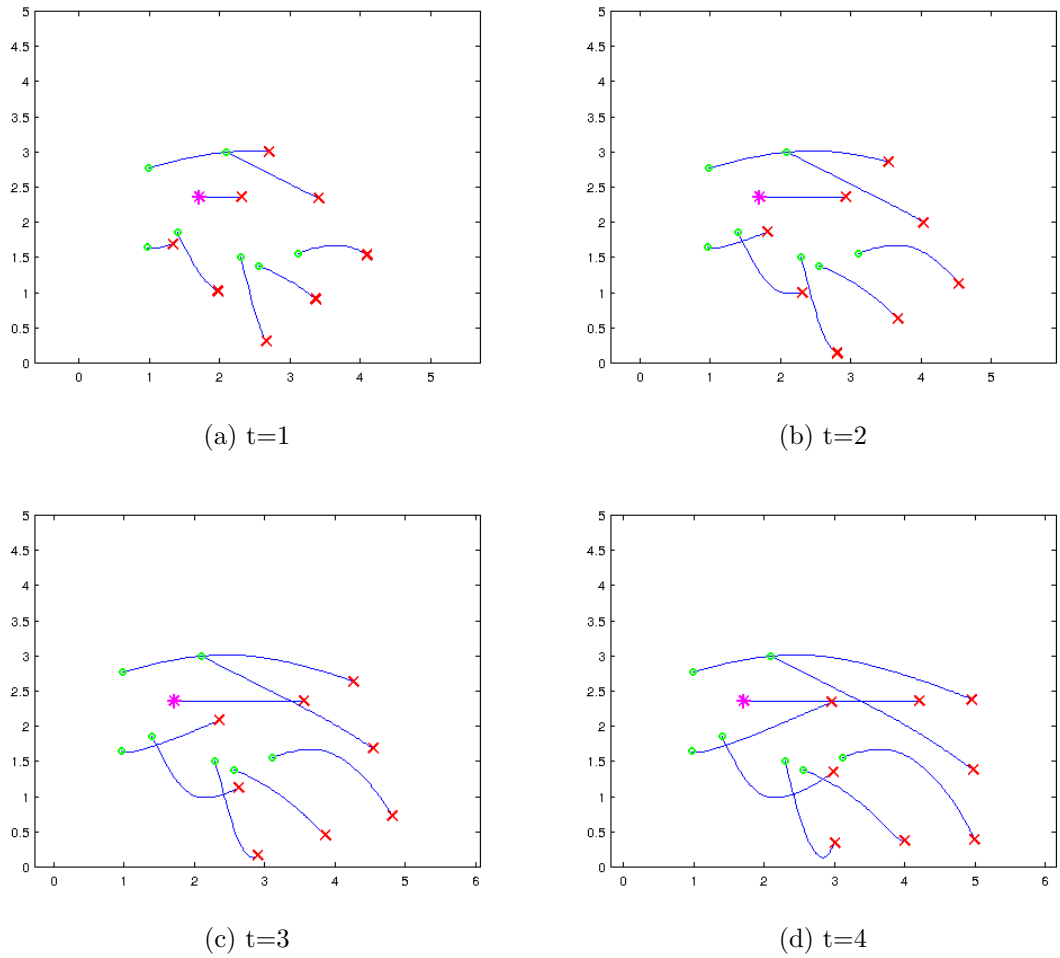
Figure 11: Simulation data for a network of 8 robots moving to a box formation with designated leader. The '*' is designated the leader and does not move. Other robots move in straight lines into formation.

4.2 Leader Based, Complete Graph Simulations

The impetus for leader based formation control is discussed in section 3.3.2. The algorithm is modified slightly such that one robot is designated the leader and has a predefined assignment. Furthermore, other robots are able to recognize the leader and constrain the formation pose and assignment so that the leader's cost for building the formation is always zero. These changes were made to the algorithm in the Matlab simulation and the tests were conducted on a network of 8 robots.

Figure 11 shows the leader based algorithm running on 8 robots attempting to build a box formation. The leader is held stationary. Note that this scenario is very similar to that shown in figure 8. However, now with the new version of the algorithm, follower robots move in straight lines to their final positions. The convergence time is also much faster with the leader version of the algorithm, although this cannot be seen in these figures. This is because the leader robot can be identified, and its position partially defines the formation pose.

Since the position of the leader defines the formation pose, we can now build the formation while the entire network is moving. The leader is free to act in any way while the followers attempt to build and maintain the formation. Figure 12 shows a



(a) t=1 (b) t=2

(c) t=3 (d) t=4

Figure 12: Formation Synthesis over time with moving leader. Leader shown by '*' moves at constant velocity to the right. Robot paths shown by lines and positions at various times shown by x's.

network of 8 robots building the box formation while the leader moves at a constant rate. Starting at t=0, the leader moves in the positive X direction at a constant rate. The follower robots successfully build and maintain the formation, even as the formation pose translates and rotates in reaction to the leader's motion. The state of the network is shown at four different times.

4.3 Non-Complete Graph Simulations

A mathematical description of non-complete graph assignment and formation control proved elusive, so simulation results served as the primary method of evaluation for

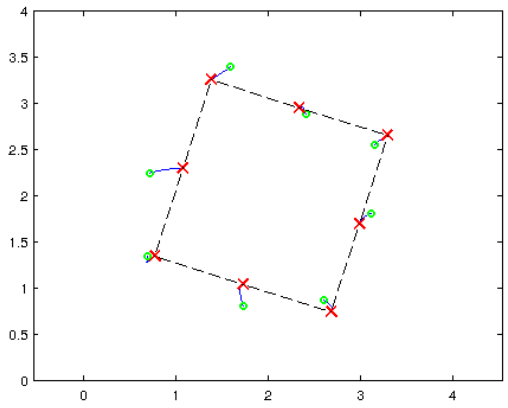


Figure 13: Simulation data for a network of 8 robots moving to a box formation with a cycle graph. Robots start sufficiently close to the desired formation such that formation synthesis is successful. Graph edges shown by dotted lines.

this case. The algorithm was modified as per the approach given in section 3.3.1. Each node uses 2N initial guesses in an attempt to find the optimal assignment given its neighbor positions. As discussed in section 3.3.1, we would expect formation synthesis to be successful if nodes start close to the desired formation. Figure 13 shows a network of 8 nodes attempting to build a box formation with only a cycle graph. Nodes can only observe the positions of their neighbors. Neighboring nodes are connected with dotted lines in the figure. Since all nodes independently come to a consistent assignment, the behavior is very similar to the complete graph case. This shows, at least empirically, that the control algorithm is locally asymptotically stable. This means it could be a viable distributed control law if only local stability is required. One potential application of this result is the ability to have time varying formations. If nodes start in a known state, that state could be used as the initial formation model. The model could then be varied over time to achieve any other desired formation shape. As long as the formation model is varied slowly enough, the nodes will always be close to the desired formation and thus the system will be asymptotically stable.

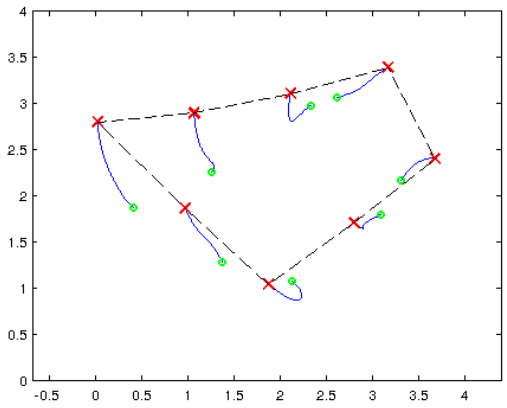


Figure 14: Simulation data for a network of 8 robots moving to a box formation with a cycle graph. Robots do not start sufficiently close to the desired formation and fail to build the formation. Graph edges shown by dotted lines.

If nodes do not start sufficiently close to the desired formation they may not independently converge to a consistent assignment. Figure 14 shows what happens when there is some contradiction in robot assignments. Notice one section of the formation is successfully assembled, yet the other half has collapsed. Robots continue to move in an attempt to reach an equilibrium point, but they move in counterproductive ways so the entire system becomes unstable.

To further investigate the properties of the non-complete graph case, we removed the assignment aspect of the algorithm to observe the behavior of the control law alone. In this case, the robot assignments are fixed, and robots can observe the assignments of their neighbors. This scenario is identical to that of distance based formation control where the assignment is known, but formation translation and rotation are unknown. This way, each robot is simply setting its target position according to the optimal translation and rotation of the formation given by equations 9 and 12 based on its neighbor set. Surprisingly, this control law appears to be stable with as little as a cycle graph, which is a flexible graph, provided there are no long range edges. Given the cycle graph below, the formation control law is stable for building the box formation regardless of the initial state! This is in contrast to the

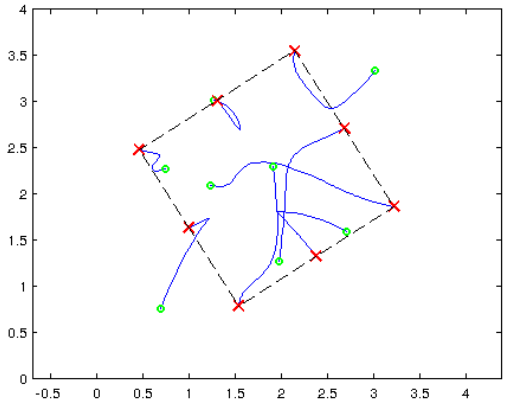


Figure 15: Simulation data for a network of 8 robots moving to a box formation with a cycle graph. Robots have fixed assignments and can recognize each other. Formation synthesis is successful for an arbitrary starting configuration. Graph edges shown by dotted lines.

distance based control method where a rigid graph is necessary in addition to initial conditions somewhat close to the desired formation. One such simulation is shown in figure 15.

Since the formation control law appears to be stable in this case, it suggests the reason for the failure in figure 14 is due to inconsistent assignment. Thus if robots start sufficiently close to the desired formation such that they independently converge to a consistent assignment, we can be reasonably confident that formation synthesis will be successful.

4.4 Summary

The simulation results presented above verify our theoretical conclusions for the complete graph cases, and give some additional insight into the non-complete graph case. The simulations suggest that the assignment algorithm and control law are globally stable for the complete graph case, and locally stable for the non-complete graph case. We have also verified that robots move in straight lines to their destinations with few intersecting trajectories for the undisturbed complete graph case. It should be noted that these simulations do not take into account many considerations that

will arise in any real world implementation such as sensor noise, actuator saturation, and vehicle dynamic constraints. Validation of the algorithm in the presence of these non-idealities is done with a physical implementation on actual robots as discussed in the following Experimentation chapter.

CHAPTER V

EXPERIMENTATION

Since the algorithm proved to be reliable in simulation, it was desirable to verify its performance on a physical system. Both the complete graph case and leader variant were tested. The distributed version was not tested since its stability properties are uncertain. The experiment consisted of using a team of 15 Khepera III robots in conjunction with a Vicon 3-D motion capture system to provide localization information. The algorithm ported over quite easily from Matlab simulation to embedded implementation on the Kheperas. We were able to successfully demonstrate the complete-graph algorithm running on all 15 Khepera III robots simultaneously. We were also able to successfully demonstrate the leader-based variation on a network of 5 robots. This chapter details the equipment used and methodology of the experiment. A presentation and discussion of the results follows.

5.1 Experimental Setup

5.1.1 Equipment

To execute the assignment and formation control algorithm requires robots with three fundamental capabilities. First, the robots must be mobile and capable of moving in a plane. Second, the robots must be able to observe the positions of all the other robots. Lastly, the robots must have sufficient computational ability to be able to implement the assignment algorithm in real time. Khepera III robots provided sufficient mobility and computational ability, yet lacked the sensing capability to localize their neighbors. Thus localization was performed with a Vicon 3-D motion capture system, and the robot positions were broadcast to the Kheperas over a wifi network. This section describes the equipment used in the experiment, namely the Khepera III robots and

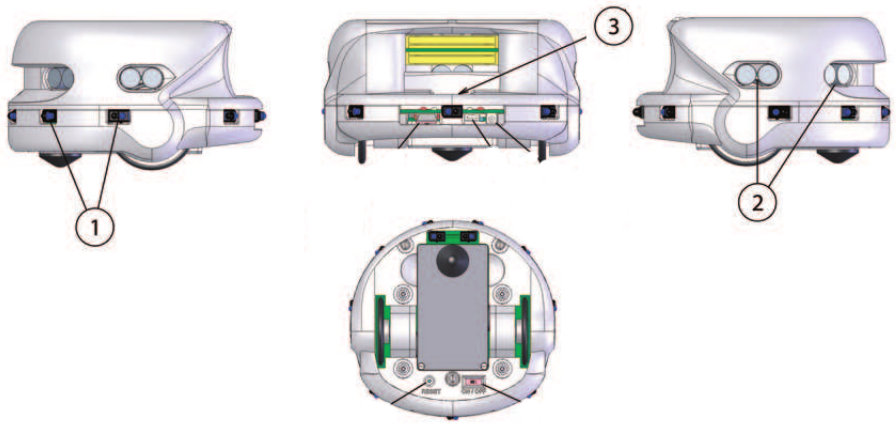


Figure 16: Illustration of the Khepera III robot. 1: Infrared Sensors, 2: Ultrasonic Sensors, 3: Expansion slot for compact flash wireless card

the Vicon motion capture system.

5.1.1.1 Khepera III Robots

The Khepera III robot is a small, modular, robotic platform specifically designed for robotic swarm type experiments. The robot is driven via two wheel differential drive with a sliding caster. The wheel drive motors contain embedded encoders which can be used for closed loop velocity control and odometry. The robot base includes an array of 9 Infrared Sensors for obstacle detection as well as 5 Ultrasonic Sensors for long range object detection [1]. The Ultrasonic sensors were not used in this experiment. Additionally, the KoreBotII module was used to provide a complete embedded Linux operating system, and an 802.11g compact flash wireless card gives access to the local area network. Figure 16 illustrates the basic physical design of the Khepera III robots. Figure 17 shows a picture of an actual Khepera robot.

Application development for the Kheperas can be programmed in C or C++ and compiled externally using the GNU cross compiler. Interfacing with one or more of

Figure 17: Picture of a Khepera III robot with a mechanical pencil shown for scale. The reflective spheres on top of the robot are used in conjunction with the Vicon motion capture system.

the Kheperas is typically done via SSH over the local area network. Each robot is assigned a unique IP address and Node ID number which are hard coded into the file system on board the robot.

5.1.1.2 Vicon Motion Capture System

The Georgia Robotics and Intelligent Systems Lab (GRITS lab) contains a Vicon motion capture system capable of tracking the 3-dimensional pose of multiple bodies simultaneously. The system operates by tracking small spherical retroreflectors using a set of 8 cameras. Retroreflectors reflect light back in the direction of the source with minimal light scattering. The same principle is used on reflective athletic gear and road signs. Each camera is surrounded by LEDs that emit light of a particular wavelength. This light is then reflected by the markers directly back towards the camera. The set of 8 cameras are calibrated so their precise relative locations are known. This way, if a marker is seen by at least two cameras its precise location in 3-D space can be calculated. Markers can be localized with an accuracy $\pm 5mm$. Figure 18 shows several of the Vicon cameras overlooking a group of Khepera robots.

Figure 18: Image of Vicon cameras overlooking a group of Khepera III robots. 3 cameras shown, 8 cameras total.

Three to four of the markers are placed on each object to be tracked, which in our case are Khepera robots (see figure 17). The markers are placed in such a way that each robot contains a unique marker pattern. This way, if its markers are seen by the cameras, the Vicon software can determine the unique robot ID in addition to its 3-D pose (x,y,z,roll,pitch,yaw). If a robot is not seen by the cameras, or if the software fails to recognize its marker pattern, it will be reported to be at position $x = y = z = 0$, $roll = pitch = yaw = 0$ by default. This provides an easy check for lost robots since the odds of a robot actually being exactly at $x = y = z = 0$, $roll = pitch = yaw = 0$ are minimal. Using this approach we have been able to track up to 15 Khepera III robots simultaneously.

5.1.2 Methods

The challenge of implementing the formation control algorithm on a real world system consisted of three primary tasks. First, the relative positions of the robots must be

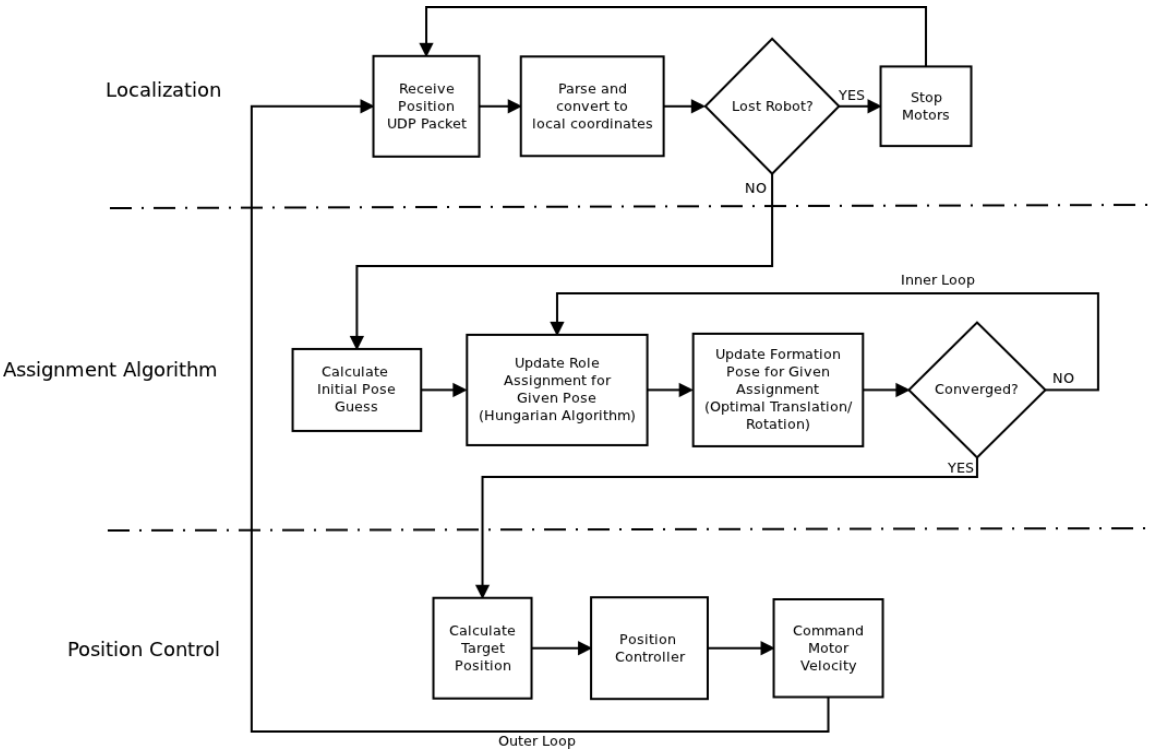


Figure 19: Software flow chart illustrating the program running on-board the Khepera robots.

known by all robots. This required transferring data from the centralized Vicon system to the individual robots. Next, the assignment algorithm had to be implemented on the embedded processor on-board the Khepera III robots. Lastly, we required a position controller that would drive the robot to the target position dictated by the assignment algorithm. Each task feeds information to the next, and the basic software flow chart is shown in figure 19. This section discusses our approach to implementing each of these three tasks.

5.1.2.1 Localization

At the start of the experiment, 15 Khepera robots are lined up in view of the Vicon cameras. Each robot has a unique marker pattern, and within the Vicon software we define which markers belong to each robot. Once configured, the Vicon software will continuously report the 3D pose of every robot that has been entered into the system.

As mentioned above, if one or more robots become no longer visible, their positions are reported to be $x = y = z = 0$, and $roll = pitch = yaw = 0$. The GRITS lab has developed custom software that interfaces with the Vicon software which reads all these robot poses and concatenates them into a single data string. The string is formatted such that each robot ID is followed by its 6 pose parameters, separated by commas. The string is then broadcast over the local area wifi network as a single User Datagram Protocol (UDP) message. This process is repeated at a rate of 5 Hz.

Meanwhile, each robot is configured to receive these UDP messages. Each robot connects to the local area wifi network with a unique, hard coded IP address and binds itself to the broadcasting socket address. After each iteration, the robot will wait for a new UDP message to arrive within a given timeout period. If a new UDP message is received, its data is parsed and each robot will take note of its own position by matching its own node ID number with that in the data string. The position data for the other robots are also stored, but the particular node IDs of other robots are ignored. A quick check is made to see if any robot positions are reported to be at $x = y = z = roll = pitch = yaw = 0$, indicating a lost robot. In this case all robots will stop moving until the Vicon system is able to find all configured robots. This is done so the system will not start to "run away" if the Vicon system fails to track the robots.

Each robot operates using its own local coordinate frame. This frame is defined such that its origin is centered on the robot, the x-axis is in the forward direction of travel, and the y-axis is to the port side as shown in figure 20. After each UDP message is parsed, each robot applies a coordinate transformation to the positions of the other robots. This converts them from the global Vicon coordinate frame, to the robot's local coordinate frame. This transformation was done for two reasons. First, it was desirable to show proper operation of the algorithm even when each robot used a unique coordinate system, which would be the case if robots localized

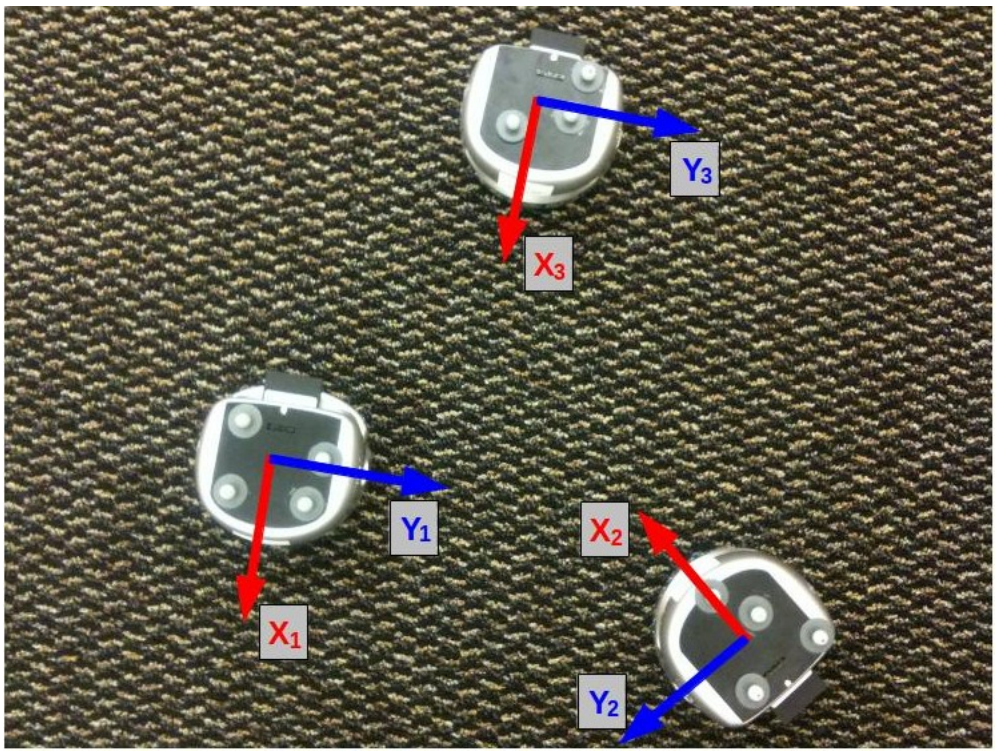


Figure 20: Overhead view showing the orientation of the robot coordinate frames.

each other with on-board sensors. Second, when target positions are given in local coordinate frames it is very easy to efficiently integrate them with a position controller as discussed in section 5.1.2.3.

5.1.2.2 Implementation of the Assignment Algorithm

With the relative position information available, the next task is to execute the assignment algorithm. This simply required porting over the Matlab code from the simulation, to a C implementation that could run on the embedded Linux processor. All together, the algorithm is comprised of three steps as illustrated in figure 19: generate initial pose guess, role assignment update, and formation pose update. Each step requires only basic arithmetic so a C implementation was relatively straightforward.

Generation of the initial pose guess is done using the method described in section 3.2.3.1. During the first iteration of the program, the initial translation guess is the

difference between the centroid of the node positions and the centroid of the formation model. The initial rotation guess is determined by aligning the eigenvectors of the moment distribution matrices. Two initial rotation guesses are needed because the eigenvectors could be either parallel or anti-parallel. This is easily implemented on an embedded processor since it only requires finding the eigenvectors of 2x2 matrices. After the first execution of the outer loop shown in figure 19, we will have obtained an initial assignment and formation pose solution. This result is then used as the initial guess during the next iteration of the outer loop. By doing this, we are guaranteed to reduce the cost function with each outer loop iteration if the robots move according to the control law as described in section 3.2.3.

If, however, the robots do not move as expected, and the cost function is found to increase from one outer loop iteration to the next, a new set of initial guesses is calculated using the moment distribution matrices. This allows for greater adaptation if there are disturbances acting on the system. It also helps guarantee that robots will continue to converge to the same assignment result independently. If robots converge to different assignment results, then the formation cost function will eventually increase from one iteration to the next. At this time, the robots will discard their previous results and use a new set of initial guesses based on the moment distribution matrices. Since all robots have access to the same data, they should then calculate the same initial guess and therefore converge to the same assignment result.

Implementation of the Hungarian Algorithm on embedded processors is nothing new. The Hungarian Algorithm is used in a wide variety of applications, and its open source implementation is readily available. We used an open source package provided under the GNU General Public License by Brian Gerkey at the University of Southern California.

Lastly, the pose update phase consists of evaluating the summations given by equation 8. This yields the optimal formation translation and rotation for the given

assignment. Then, the formation model points are translated and rotated using these parameters. The inner loop then jumps back to the Hungarian Algorithm phase using the newly transformed formation points. This process repeats until convergence. Once we achieve convergence, we have a final formation pose and assignment which is fed into the position controller.

5.1.2.3 Position Control

Given the final formation pose and role assignment, a target position for the robot can be found by selecting the appropriate transformed formation model point. Since everything is done in the robot coordinate frame, this gives us a relative target position to drive towards. The role of the position controller is to calculate the appropriate left and right motor speeds that will drive the robot towards its target. If the target position is fixed in the global coordinate frame, then the robot should asymptotically approach its target.

With a differential drive providing locomotion, the Kheperas can be regarded as a "Unicycle" type robot. This means at any instant in time we can control our forward velocity, v, and rotational velocity, w. For ease of analysis, we will temporarily revert to using to using the global coordinate frame, then apply a coordinate transform to express the control law in terms of the robot's local coordinate system. We are also assuming that the robot can accelerate fast enough so that the robot's inertial dynamics can be ignored. The following control law was developed by Peter Kingston and Jean-Pierre de la Croix in the Georgia Robotics and Intelligent Systems lab:

$$\begin{aligned} w = &\quad -k_1 \, P^T \, J \, U \\ v = &\quad -k_2 \, P^T \, U \end{aligned} \tag{39}$$

Where $P = [p_x \; p_y]^T$ is the robot's position relative to the origin, $U = [u_x \; u_y]^T$ is a unit vector representing the robot's heading, k_1 and k_2 are control gains, and J is the

90^o rotation matrix. v, and w are the commanded linear and rotational velocities, respectively. This control law will asymptotically drive the robot to $P = [0\ 0]^T$.

Proof. The position and heading of the robot change over time according to:

$$\begin{aligned} \dot{P} &= v\,U \\ \dot{U} &= w\,J\,U \end{aligned} \tag{40}$$

If we choose a candidate Lyapunov function to be: $V = \frac{1}{2}P^T P$, then:

$$\dot{V} \quad = \quad -k_2\,(P^T\,U)^2 \quad \leq \quad 0 \tag{41}$$

$\dot{V} = 0$ only if $P^T\,U = 0$. However if $P^T\,U = 0$, then $P^T\,J\,U = \|P\|$. So if $\|P\| \neq 0$ and $k_1 \neq 0$, then $w \neq 0$ and therefore, U is time varying. Hence, by LaSalle's Invariance Principle, asymptotic stability is assured. □

It is now possible to perform a coordinate frame transformation to the local robot coordinate system without affecting the stability properties of the controller. The position vector, P, can be expressed in the robot coordinate frame as:

$$P' = \begin{bmatrix} -u_x & u_y \\ -u_y & -u_x \end{bmatrix} P \tag{42}$$

Where P' is the vector P expressed in the robot local coordinate frame. Substituting P' into equation 39 yields:

$$w = \quad -k_1\,P'^{\,T} \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{43}$$

$$v = \quad -k_2\,P'^{T} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{44}$$

$-P'$ can be interpreted as the vector pointing from the robot to its target position, expressed in the robot coordinate frame. Thus, the linear velocity, v, can simply be

set proportional to the x component of the target position expressed in local robot coordinates. Similarly, the rotational velocity, w, can be set proportional to the y component of the target position. This will cause the robot to approach its target position asymptotically. This position controller was implemented on the Khepera robots.

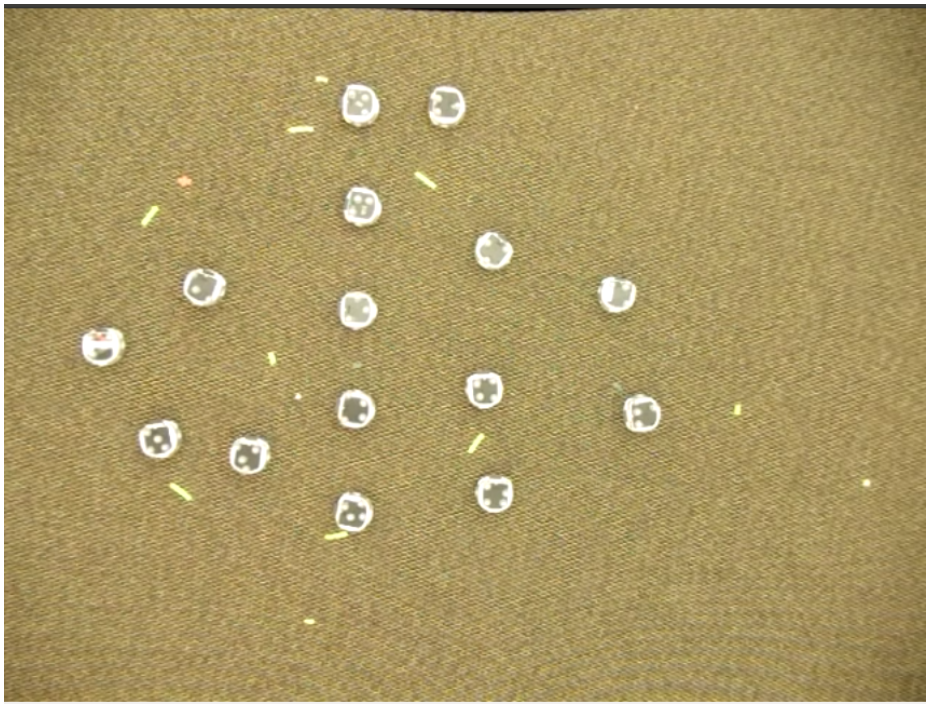
5.2 Experimental Results

Using the equipment and methods described above, the assignment algorithm was successfully demonstrated with up to 15 Khepera robots. First, the standard complete graph version was tested where each of the 15 robots executed the same program. Then, the leader based version was run on 5 robots where one robot was designated the leader and acted independently. Both cases were successful in efficiently creating the desired formation. This section presents the details and a video of each test, followed by a discussion of the results.

5.2.1 GRITS Formation Sequence

The standard, complete graph version of the assignment algorithm was tested by executing a sequence of formations on 15 Khepera robots. The chosen formations were in the form of the letters G,R,I,T,S (for Georgia Robotics and InTelligent Systems lab). Each formation model was defined by a set of points stored in a text file on board each robot. All robots are given the names of the formation files to be used in the sequence. Each robot knows the formation has been built when the cost function, given by equation 5, approaches zero. This way we perform a formation sequence by each robot indexing to the next formation file when the cost function goes below a predefined level.

Figure 21 contains a link to a video of this experiment. The robots start in a random configuration when they are commanded to move to the first formation in the sequence, "G". No collision avoidance is used as we are relying on the fact that



Click to view macdonald-edward-a-201108-mast-formation-15.avi

Figure 21: A video of 15 Khepera robots demonstrating the assignment algorithm using a sequence of formations to spell "G-R-I-T-S" (links to *macdonald-edward-a-201108-mast-formation-15.avi*)

robot trajectories rarely intersect, so collisions are unlikely. Although, we did use the Khepera front facing infrared proximity sensor to stop the motors if an obstacle is detected directly in front of the robot.

The assignment algorithm behaves as expected and the robots successfully complete the formation sequence. Once the sequence was complete the robustness to unexpected disturbances was tested. As seen in the video, if a single robot is picked up and placed at a new location, the remaining robots re-evaluate their assignments and choose a new assignment with a lesser cost. This is repeated three times at various locations. Next, a single robot is picked up, pulled out of formation, and held still so it cannot move. The video shows the remaining robots altering the formation pose to reduce the formation cost. Notice the remaining robots are moving very slowly towards the constrained robot.

5.2.1.1 Observations and Discussion

This implementation of the assignment algorithm performed well during this experiment and very closely matched the results found in simulation. However there were two main, observable differences between this implementation and simulation. Firstly, the Khepera robots have actuation limits and have limited velocities before the motors become saturated. In this scenario, robots are never very far away from their target positions so this effect is relatively negligible. Secondly, and more significantly, is the fact that these robots are non-holonomic and cannot drive instantaneously in any direction. This means that robots cannot move in straight lines to their destinations, but may have to perform "3 point turns" before driving towards their target. Thus robots must deviate from the optimal path, which would be a straight line. Most of the time this is not much of a concern, however occasionally it so happens that by deviating from the optimal path, a new assignment is found that has lower cost than the original. This is the case when robots begin moving to build the formation, but then suddenly switch directions. This phenomenon can be observed at time 0:59 of the video in figure 21.

5.2.2 Leader Based Formation Control

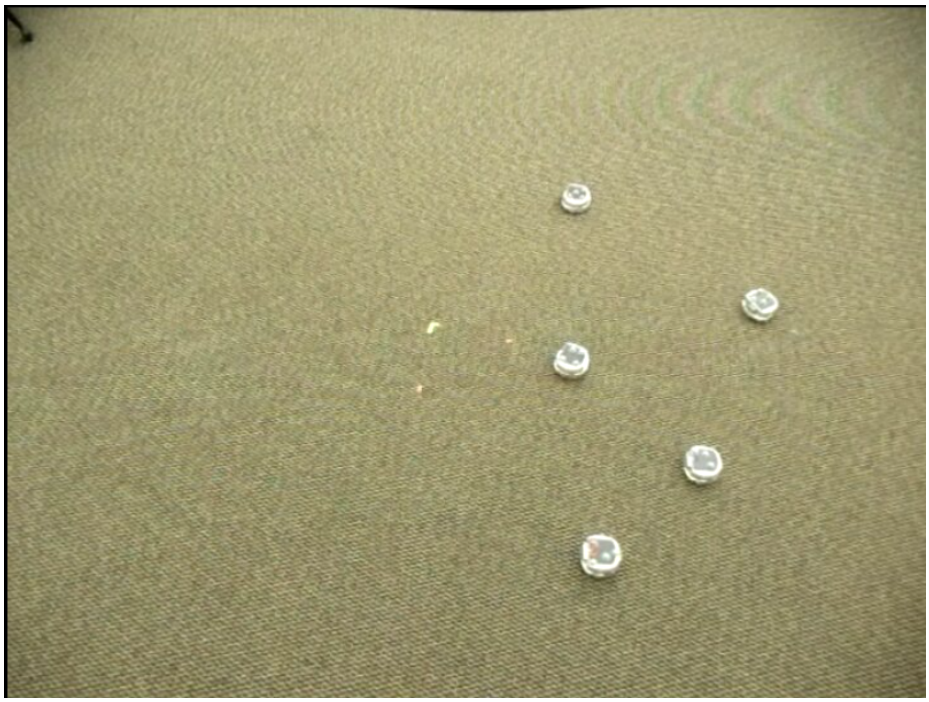
Following the success of the formation sequence experiment with 15 robots, we decided to attempt to implement the leader based version of the assignment algorithm as well. This required only minor modifications to the program on the Kheperas. It simply involved constraining the role assignment and formation pose as discussed in section 3.3.2. This revised version of the program was placed on the follower robots. The leader robot was programmed to continuously drive in an ellipse around the perimeter of the room, independently of the actions of the follower robots. It was not necessary to modify the formation model files in any way. The robots are no longer

programmed to automatically index to the next formation, rather formation transitions are commanded manually. For this scenario we used only four follower robots and one leader robot. Using all 15 robots would have created too much congestion as they attempt to build the formation while driving around the room.

Figure 22 contains a link to a video of this experiment. First the leader is commanded to begin its elliptical circuit around the room. Then, the follower robots are commanded to move into the first formation, a pentagon. At time 0:30 of the video, the robots are commanded to move into a line formation. At time 0:45 the follower robots are commanded to move into a tight box formation offset from the leader. At time 0:52 the followers are commanded to return to the pentagon shape. At time 0:59 robots are commanded to build a large box formation centered about the leader. At 1:17 the robots again form a pentagon. At 1:27 the robots return to the line formation. At 1:35 robots are disturbed to demonstrate reassignment while moving. Starting at 1:42 robots are disturbed to demonstrate modification of the formation pose. Lastly, at 2:08 the leader is stopped and the system is disturbed further to demonstrate how the follower robots react.

5.2.2.1 Observations and Discussion

This experiment successfully demonstrated the execution of the leader based version of the assignment algorithm. Robots demonstrated the ability to modify the formation pose, and dynamically assign themselves while the formation was in motion. The algorithm performed very similarly to the simulation, however non-idealities began to play a larger role than the stationary formation case. Actuation limits are no longer negligible since robots are now attempting to reach a moving target requiring them to move at higher speeds. At many points in time the follower robots are moving at their maximum speeds. If the leader robot is made to move too fast, it is possible for the follower robots to never reach their position targets. Robot



Click to view macdonald-edward-a-201108-mast-formation-leader.avi

Figure 22: A video of 5 Khepera robots demonstrating the leader version of the assignment algorithm using several geometric formations (links to *macdonald-edward-a-201108-mast-formation-leader.avi*)

dynamics also begin to affect the behavior since high accelerations are required to react to moving position targets. Inter-robot collisions also became an issue since no additional collision avoidance behaviors were added. The assumption that robot trajectories rarely intersect is no longer valid since the leader robot is following an independent trajectory. The choice of formation transitions was deliberate to prevent follower robots from colliding with the leader. The addition of a simple collision avoidance behavior could prevent this issue.

One additional result, evident in the video, is the fact that the exact desired formation is never achieved unless the leader is stationary. For example, at time 0:59 in the video, the robots should make a box centered about the leader. Clearly the leader has some offset from the center of the box, in the direction of travel. This is expected since the robot position controller has no integral term. The followers must accumulate some amount of error between their target positions and actual positions in order to generate a velocity command. The steady state formation error will be proportional to the leader's velocity. This could be alleviated by the addition of a feed forward velocity term, or an integral term, in the position controller.

5.3 Summary

By using 15 Khepera III robots, and a Vicon motion capture system, we were able to successfully demonstrate the distributed implementation of the assignment algorithm. While we relied heavily on the overhead motion capture system, a more real world application could use GPS to localize each robot. Each robot could then broadcast its own position to the others. The non-idealities in the system did have some impact on algorithm performance, however these effects were minor and formation synthesis was achieved. The algorithm proved to be robust to disturbances to the system, and dynamic changes to the formation pose and role assignment were observed. The standard complete graph assignment algorithm was verified with 15 robots, and the

inherent collision avoidance assumption was validated. The leader based version of the assignment algorithm was verified with 5 robots and demonstrated the ability to dynamically assign roles while the formation pose was in motion.

CHAPTER VI

CONCLUSION

As multi-robot solutions to real world problems become viable in the 21st century, new methods of coordinating these robots become necessary. With applications in areas ranging from medical nano-robots to space exploration, interest in the control of these multi-robot systems will continue to grow. Assignment and formation control is a fundamental component of many of these networked systems, and has been a focus of this research.

This thesis has presented a new algorithm to solve the problem of assigning mobile agents to roles within a translationally and rotationally invariant formation. It specifically applies to the problem of finding an efficient assignment when the formation pose is initially unknown. When implemented in parallel on a network of mobile robots, it offers some advantages over previous methods including: the use of independent robot coordinate frames, dynamic re-assignment for changing conditions, and inherent collision avoidance. In addition, robots do not need to identify the assignments of their neighbors. The greatest drawback to the assignment algorithm is that it requires a complete graph for guaranteed formation stability. Simulation evidence shows the assignment algorithm can work on non-complete graphs, but sufficient conditions for stability are unknown. A variation of the algorithm was presented for leader based formation control that introduced the ability for robots to assign themselves and build formations while the formation is in motion.

The assignment algorithm was demonstrated using 15 Khepera III robots and a motion tracking system. While the non-idealities of the real world system have some

minor impact on performance, the algorithm was successful in building stable formations and demonstrated the ability to dynamically re-assign roles. An experiment with the leader based version of the assignment algorithm also demonstrated the ability to assign and construct moving formations based on the position of a leader.

6.1 Future Work

This thesis has been primarily focused on the development of the assignment algorithm when it is applied to network of mobile robots that have complete information about the positions of all other robots (the complete graph case). It has been shown that the algorithm is guaranteed to converge, and that formations will be asymptotically stable. However, there remain many unanswered questions regarding the non-complete graph case. In simulation, it has proven to be a viable method of assignment and formation control, but sufficient conditions for building stable formations are still unknown. The distributed version of the algorithm would be much more useful if it could be determined what graph topology is required, given the robot positions, for the robots to independently reach a consistent assignment.

The algorithm could be further augmented with the inclusion of inter-agent communication or node labels. If nodes can make their particular choice of assignment known to their neighbors, perhaps the algorithm could somehow be augmented or constrained to take this information into account. This may give a means of developing a truly distributed assignment algorithm, although the best approach to this problem is unclear at this point.

Another intriguing observation, which resulted from simulation, was that if the assignments are fixed and known in the non-complete graph case, stable formations can be built even if the graph is flexible. This is a result of each agent simply estimating the optimal pose of the formation based on its neighbor set. This could be a viable alternative to distance based formation control when graphs are non-rigid.

However a stability proof, and sufficient conditions for stability, are lacking. If the stability of such a control law could be proven, this could be a significant result.

This algorithm has shown that methods from computer perception can be successfully applied to problems in networked control systems. Identifying one's role within a network is very much a perception problem, and surely other concepts from machine perception could be applied. An open area for future research is the study of what other machine perception algorithms could be used in the estimation of global network parameters by using only local information.

REFERENCES

[1] "K-team corporation website." http://www.k-team.com/, May 2011.

[2] Arthur, D. and Vassilvitskii, S., "Worst-case and smoothed analysis of the icp algorithm, with an application to the k-means method," *IEEE Symposium on Foundations of Computer Science*, 2006.

[3] Baillieul, J. and Suri, A., "Information patterns and hedging brockett's theorem in controlling vehicle fromations," *IEEE Conf. on Decision and Control*, vol. 42, Dec. 2003.

[4] Bertsekas, D. P. and Castanon, D. A., "Parallel synchronous and asynchronous implementations of the auction algorithm," *Parallel Computing*, vol. 17, pp. 707–732, 1991.

[5] Besl, P. J. and McKay, N. D., "A method for registration of 3-d shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, Feb. 1992.

[6] Chen, Y. and Medioni, G., "Object modeling by registration of multiple range images," *International Conference on Robotics and Automation*, April 1991.

[7] Couzin, I. D., Krause, J., Franks, N. R., and Levin, S. A., "Effective leadership and decision-making in animal groups on the move," *Nature*, vol. 433, Feb 2005.

[8] Fitzgibbon, A., "Robust registration of 2d and 3d point sets," *Image and Vision Computing*, vol. 21, pp. 1145–1153, 2003.

[9] Jadbabaie, A., "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Trans. Autom. Control*, vol. 48, pp. 988–1001, Jun. 2003.

[10] Kowalczyk, W., "Target assignment strategy for scattered robots building formation," *International Workshop on Robot Motion and Control*, November 2002.

[11] Kuhn, H. W., "The hungarian method for the assignment problem," *Naval Research Logistics*, vol. 52, pp. 7–21, 1955.

[12] Lambercy, F. and Caprari, G., *Khepera III manual ver 2.2.* K-team.

[13] Lemay, M. and Michaud, F., "Autonomous initialization of robot formations," *International Conference on Robotics and Automation*, April 2004.

[14] Mesbahi, M. and Egerstedt, M., *Graph Theoretic Methods in Multiagent Networks*. Princeton University Press, 2010.

[15] Michael, Zavlanos, and Pappas, "Distributed multi-robot task assignment and formation control," *IEEE International Conference on Robotics and Automation*, May 2008.

[16] Munkres, J., "Algorithms for the assignment and transportation problems," *Journal of the Society for Industrial and Applied Mathematics*, vol. 5, pp. 32–38, March 1957.

[17] Oflati-Saber, R. and Murray, R., "Graph rigidity and distributed formation stabilization of multi-vehicle systems," *IEEE Conference on Decision and Control*, December 2002.

[18] Oh, K.-K., "A survey of formation of mobile agents," *IEEE International Symposium on Intelligent Control*, September 2010.

[19] Olfati-Saber, R. and Murray, R., "Consensus protocols for networks of dynamic agents," *American Control Conference*, June 2003.

[20] Zamani, M. and Lin, H., "Weights' assignment in multi-agent systems under a time-varying topology," *IEEE International Conference on Advanced Intelligent Mechatronics*, July 2009.

[21] Zavlanos and Pappas, "Sensor-based dynamic assignment in distributed motion planning," *IEEE International Conference on Robotics and Automation*, April 2007.

[22] Zavlanos, M. and Pappas, G., "Distributed formation control with permutation symmetries," *IEEE Conference on Decision and Control*, December 2007.

[23] Zavlanos, M. and Pappas, G., "Dynamic assignment in distributed motion planning with limited infromation," *American Control Conference*, July 2007.

[24] Zavlanos, M. and Pappas, G., "Dynamic assignment in distributed motion planning with local coordination," *IEEE Transactions on Robotics*, vol. 24, pp. 232–242, February 2008.

[25] Zavlanos, M., Spesivtsev, L., and Pappas, G., "A distributed auction algorithm for the assignment problem," *IEEE Conference on Decision and Control*, December 2008.